Sparse Kernel Machines

Prof. Nicholas Zabaras
Center for Informatics and Computational Science
https://cics.nd.edu/
University of Notre Dame
Notre Dame, Indiana, USA

Email: nzabaras@gmail.com
URL: https://www.zabaras.com/

April 3, 2019

SVM - Contents

- Support Vector Machines, Linearly Separable Training Data, Maximum Margin Classifiers, Constrained Optimization Problem, Dual Formulation, Quadratic Programming, KKT Conditions, Equivalent Error Function Formulation, Example – SVM with a Gaussian Kernel
- Overlapping Class Distributions, v-SVM, Chunking, Decomposition Methods, Probabilistic Predictions with SVM
- SVM and Logistic Regression, Multiclass SVMs
- SVM for Regression, Computational Learning Theory

Following closely:
Bishop CM, *Pattern Recognition and Machine Learning*, Springer, 2006 (Chapter 7)
Murphy, K. Machine Learning: A probabilistic Approach (Chapter 14)

RVM - Contents

- Relevance Vector Machines: Likelihood, Prior, Posterior, Predictive Distribution
- Evidence Approximation, Evidence, Two useful matrix identities
- Fast training algorithm
- Splitting the evidence, Analysis of the evidence
- Complete algorithm, Example
- Fast updates
- Extenstion to Multiple Outputs, Numerical Stability Concerns

Support Vector Machines

Introduction

- Significant limitation of earlier discussed algorithms is that $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ must be evaluated for all possible $\boldsymbol{x}_n, \boldsymbol{x}_m$ of training points
  - ✓ Computationally infeasible during training
  - ✓ Excessive computation when making predictions for test data.
- In this lecture, we discuss kernel-based algorithms that have *sparse* solutions
  - ✓ predictions for new inputs depend only on the kernel function evaluated at a subset of the training data points.
- In the *support vector machine* (SVM), the determination of the model parameters corresponds to a convex optimization problem, i.e. any local solution is also a global optimum.
- However, SVM does not provide probabilistic solutions.

Linearly Separable Training Data

- Consider two-class classification problem using linear models

$$y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b$$

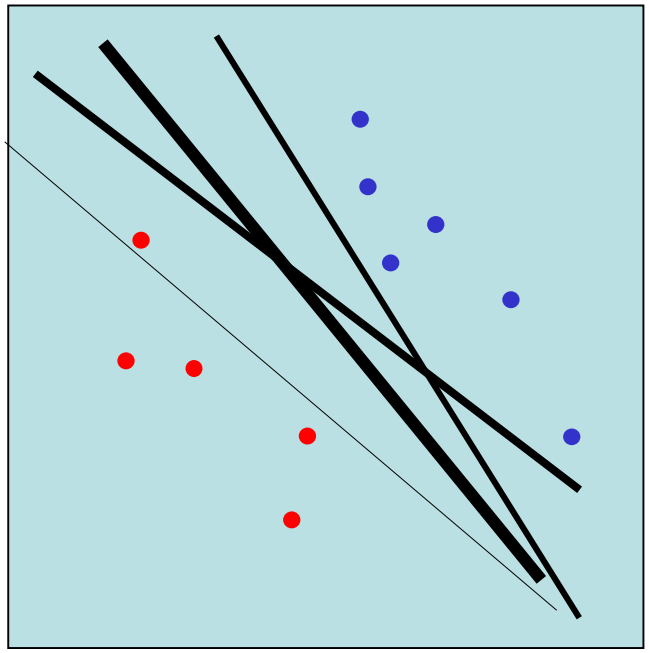
- The training data set comprises N input vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, with corresponding target values $t_1, \ldots, t_N$ where $t_n \in \{-1, 1\}$, and new data points $\boldsymbol{x}$ are classified according to the sign of $y(\boldsymbol{x})$.

- Assume that the training data set is linearly separable in feature space: there exists at least one choice of $\boldsymbol{w}$ and b such that $y(\boldsymbol{x}_n) > 0$ for points having $t_n = +1$ and $y(\boldsymbol{x}_n) < 0$ for points having $t_n = -1$, so that $t_n y(\boldsymbol{x}_n) > 0$ for all training data points.

- Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer.
- Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining* **2**(2), 121–167.
- Cristianini, N. and J. Shawe-Taylor (2000). *Support vector machines and other kernel-based learning methods*. Cambridge University Press.
- Müller, K. R., S. Mika, G. R¨atsch, K. Tsuda, and B. Schölkopf (2001). An introduction to kernel based learning algorithms. *IEEE Transactions on Neural Networks* **12**(2), 181–202.
- Schölkopf, B. and A. J. Smola (2002). *Learning with Kernels*. MIT Press.
- Herbrich, R. (2002). *Learning Kernel Classifiers*. MIT Press.

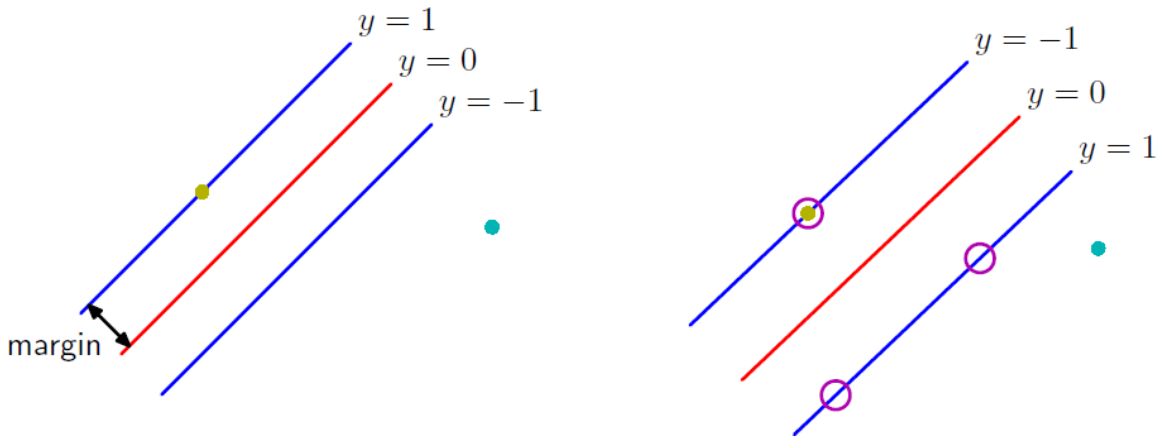
Linearly Separable Training Data

- Suppose we use a big set of features to ensure that the two classes are linearly separable. What is the best separating line to use?
  - ✓ The Bayesian answer is to use them all (including ones that do not quite separate the data.)
  - ✓ Weight each line by its posterior probability (i.e. by a combination of how well it fits the data and how well it fits the prior).
  - ✓ Is there an efficient way to approximate the correct Bayesian answer?



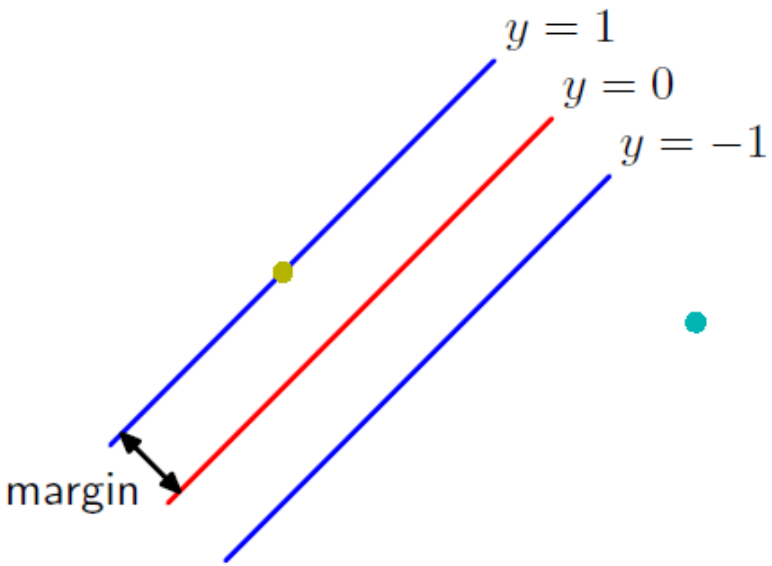
Maximum Margin Classifiers

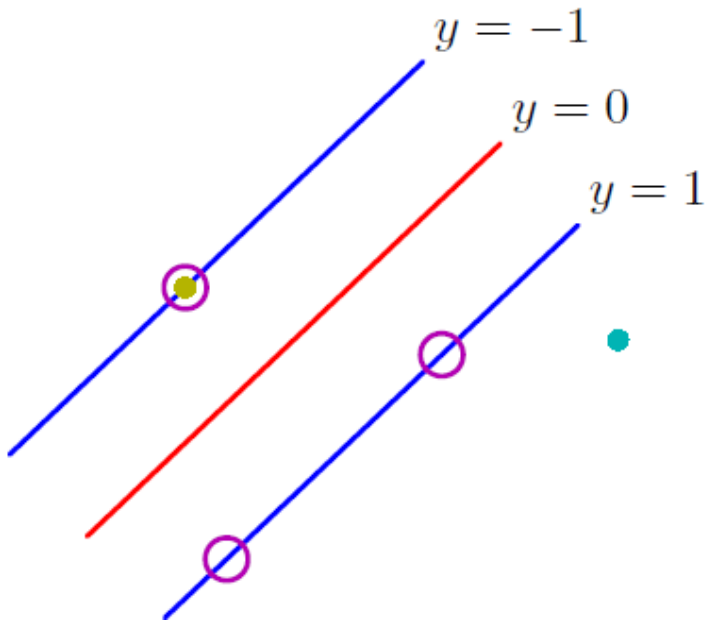
- There may exist many solutions that separate the classes exactly.
- E.g. the perceptron algorithm finds a solution in a finite number of steps that depends on the initial values of $\boldsymbol{w}$ and b and the order in which the data points are presented.
- When multiple solutions find the one that gives the smallest generalization error.
- SVM approaches this problem through the *margin* - smallest distance between the decision boundary and any of the samples.



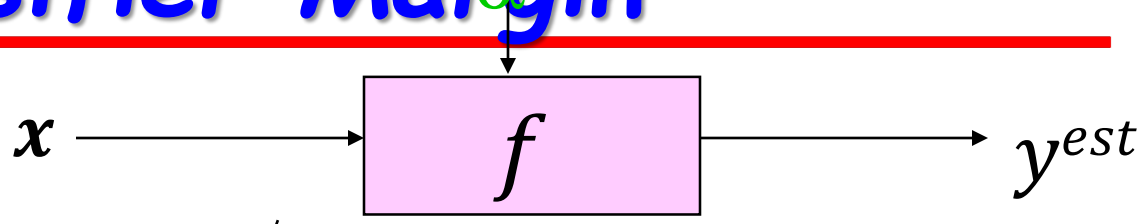
Maximum Margin Classifiers

- The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points.
- Maximizing the margin leads to a particular choice of decision boundary.
- The location of this boundary is determined by a subset of the data points, known as support vectors (circles).



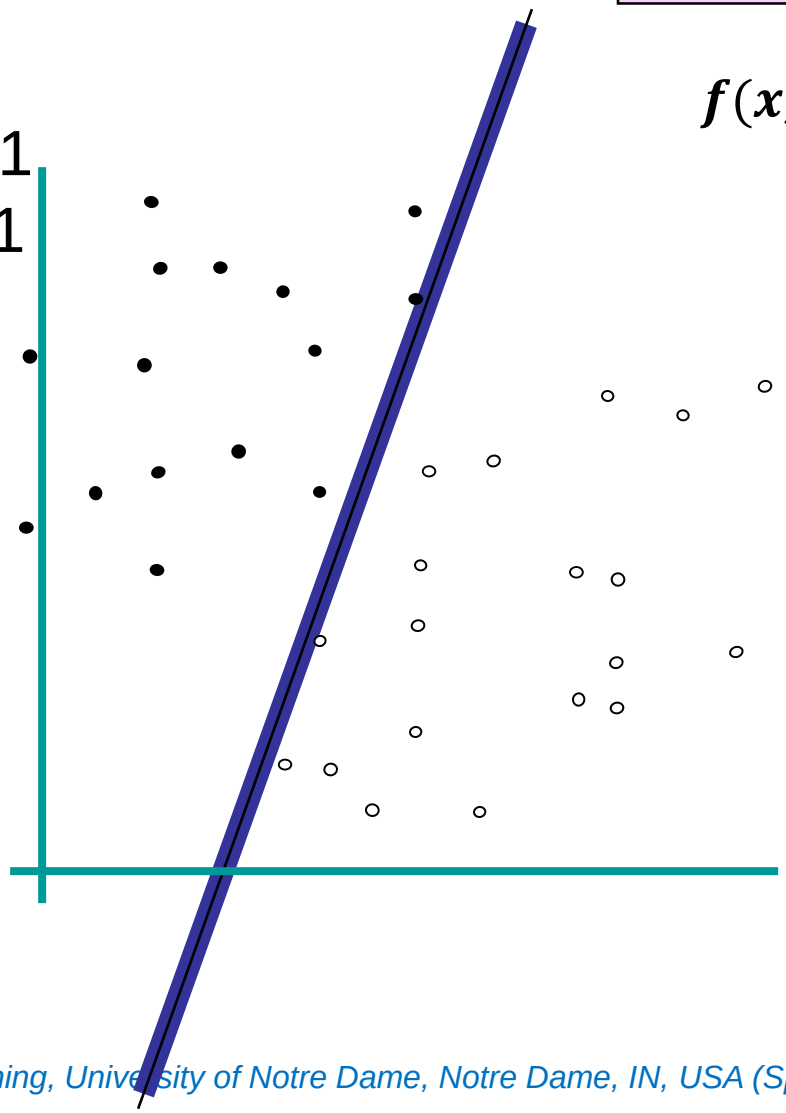


Classifier Margin



$f(\boldsymbol{x}, \boldsymbol{w}, b) = sign(\boldsymbol{w}.\boldsymbol{x} + b)$

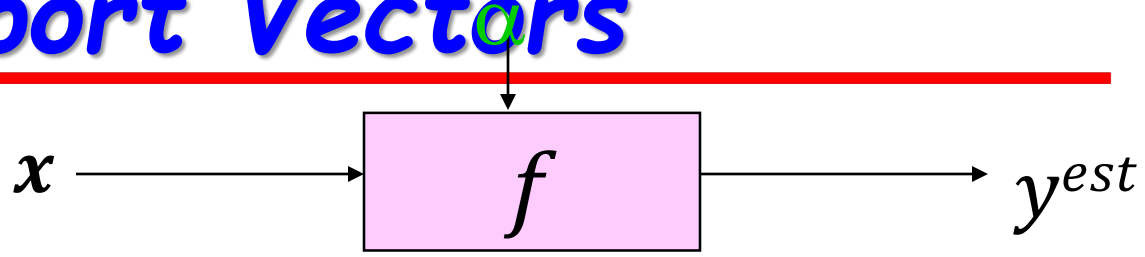
- denotes +1
- denotes -1



Linear SVM

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Support Vectors



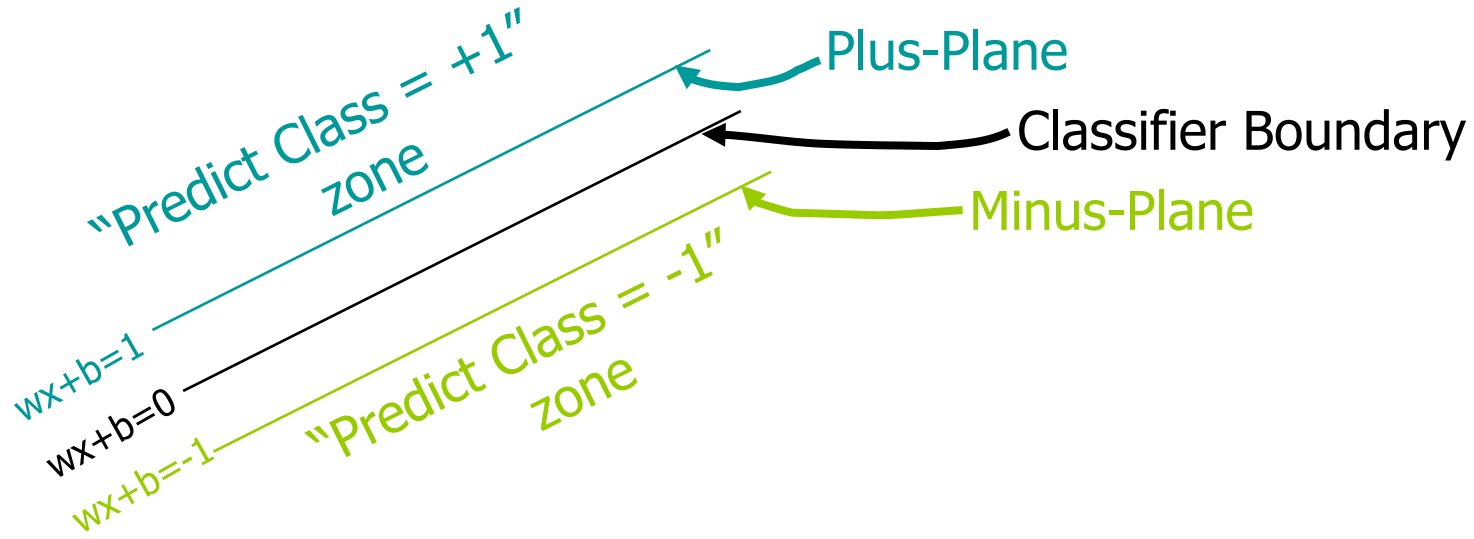
$f(x, w, b) = sign(w.x + b)$

- denotes +1
- denotes -1

Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the maximum margin.

Specifying a line and margin



- Plus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = +1 \}$
- Minus-plane = $\{ \boldsymbol{x} : \boldsymbol{w} . \boldsymbol{x} + b = -1 \}$

Classify as.. $+1$ if $\boldsymbol{w} . \boldsymbol{x} + b \geq 1$

-1 if $\boldsymbol{w} . \boldsymbol{x} + b \leq -1$

Maximum Margin Classifiers

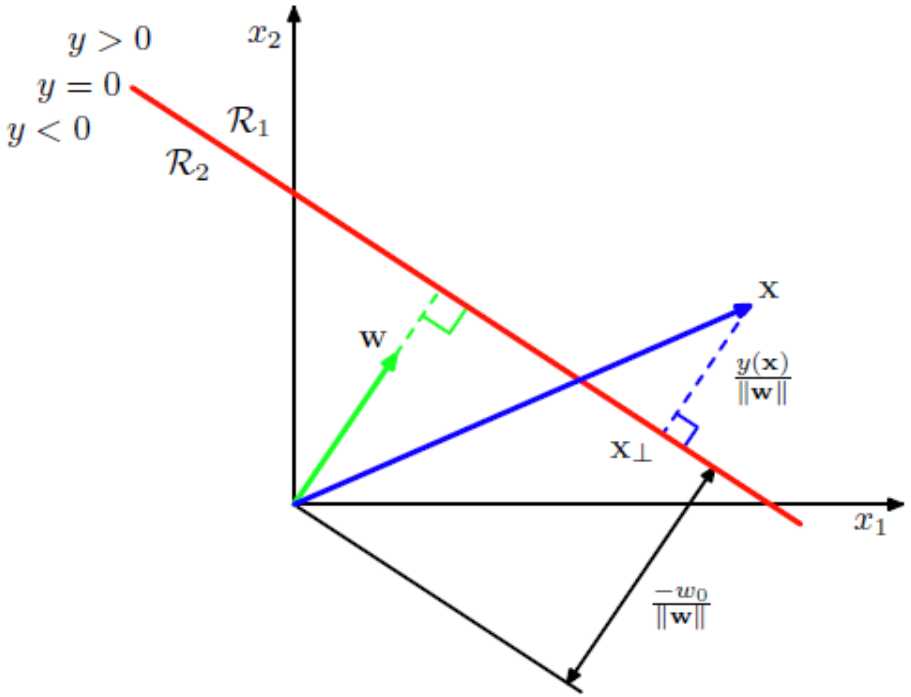
- The maximum margin can be motivated with *statistical learning theory*.
- An insight has been given by Tong and Koller (2000) who consider a hybrid (generative and discriminative) classification approach.
- They first model the distribution over $\boldsymbol{x}$ for each class using a Parzen density estimator with Gaussian kernels having a common σ^2. Together with the class priors, this defines an optimal misclassification-rate decision boundary.
- Instead of using this optimal boundary, they determine the best hyperplane by minimizing the probability of error relative to the learned density model. As $\sigma^2 \rightarrow 0$, the optimal hyperplane becomes one having maximum margin.
- As $\sigma^2 \rightarrow 0$, the hyperplane becomes independent of data points that are not support vectors.

- Tong, S. and D. Koller (2000). Restricted Bayes optimal classifiers. In *Proceedings 17th National Conference on Artificial Intelligence*, pp. 658– 664. AAAI.

Maximum Margin Solution

- Since we are only interested in correctly classified points, $t_n y(\boldsymbol{x}_n) > 0$, the vertical distance of $\boldsymbol{x}_n$ to the decision boundary $y(\boldsymbol{x}) = 0$ is:

$$\frac{t_n y(\boldsymbol{x}_n)}{\|\boldsymbol{w}\|} = \frac{t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b)}{\|\boldsymbol{w}\|}$$



- The max margin solution is obtained as:

$$\underset{\boldsymbol{w},b}{\operatorname{argmax}} \left\{ \frac{1}{\|\boldsymbol{w}\|} \min_n t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b) \right\}$$

- Scale $\boldsymbol{w}$ and b such that for the point that is closest to the surface:

$$t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b) = 1$$

- Thus for all points $n = 1, .., N$

$$t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b) \geq 1$$

- The problem now is: $\underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{w}\|^2$ under $t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b) \geq 1$

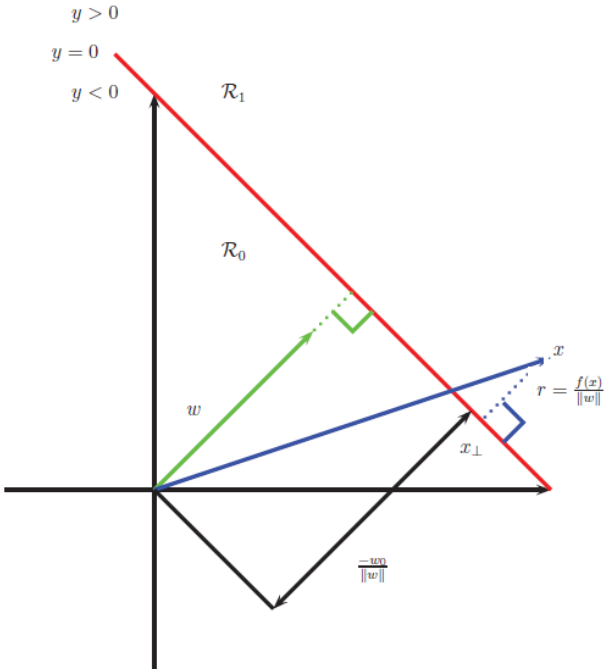
Maximum Margin Solution

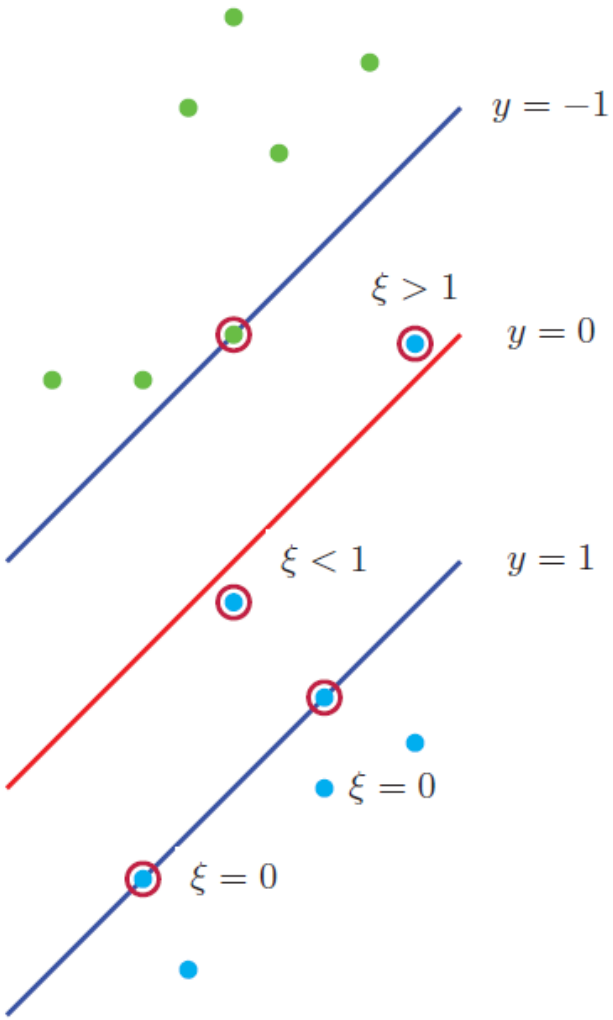
- Our problem is stated as follows:

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 \; subject\ to$$

$$t_i[b + \boldsymbol{w}^T\phi(\boldsymbol{x}_i)] - 1 \geq 0, i = 1, \ldots, N$$

(each point is on the correct side of the boundary)





Constrained Optimization

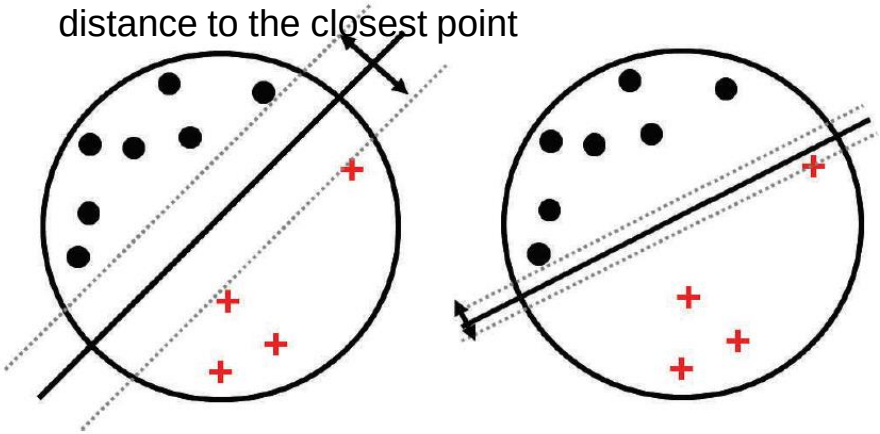
- Our problem is stated as follows:

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 \;\; subject\ to$$

$$t_i[b + \boldsymbol{w}^T\phi(\boldsymbol{x}_i)] - 1 \geq 0, i = 1, \ldots, N$$

(each point is on the correct side of the boundary)

Margin here defined as the perpendicular distance to the closest point



Large margin Small margin

- We represent the constraints with Lagrange multipliers as losses:

$$\max_{\alpha \geq 0} \alpha_i \left(1 - t_i[b + \boldsymbol{w}^T\phi(\boldsymbol{x}_i)]\right) = \begin{cases} 0, if\ t_i[b + \boldsymbol{w}^T\phi(\boldsymbol{x}_i)] - 1 \geq 0 \\ \infty, \qquad otherwise \end{cases}$$

- The minimization problem can now be stated as:

$$\min_{\boldsymbol{w}} \left\{ \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{N} \max_{\alpha \geq 0} \alpha_i \left(1 - t_i[b + \boldsymbol{w}^T\phi(\boldsymbol{x}_i)]\right) \right\}$$

SVM – Separable Case

- The minimization problem can now be stated as:

$$\min_{\boldsymbol{w}} \max_{\{\alpha_i \geq 0\}} \left\{ \frac{1}{2} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{N} \alpha_i (1 - t_i [b + \boldsymbol{w}^T \phi(\boldsymbol{x}_i)]) \right\} =$$

$$\max_{\{\alpha_i \geq 0\}} \min_{\boldsymbol{w}} \left\{ \frac{1}{2} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{N} \alpha_i (1 - t_i [b + \boldsymbol{w}^T \phi(\boldsymbol{x}_i)]) \right\}$$

- As a result we should be able to minimize wrt $\boldsymbol{w}$ for any given set of the α's the following:

$$J(\boldsymbol{w}; \alpha) = \left\{ \frac{1}{2} \|\boldsymbol{w}\|^2 + \sum_{i=1}^{N} \alpha_i (1 - t_i [b + \boldsymbol{w}^T \phi(\boldsymbol{x}_i)]) \right\}$$

- Setting the derivatives wrt to $\boldsymbol{w}$ and b gives:

$$\boldsymbol{w} - \sum_{i=1}^{N} \alpha_i \, t_i \phi(\boldsymbol{x}_i) = 0, \; - \sum_{i=1}^{N} \alpha_i \, t_i = 0$$

Constrained Optimization Problem

- Introducing Lagrange multipliers $a_n \geq 0$, our objective function is:

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n=1}^{N} a_n(t_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) + b) - 1\ )$$

- Note we minimize wrt $\boldsymbol{w}$ and b and maximize wrt $\boldsymbol{a}$. Setting the derivatives with respect to $\boldsymbol{w}$ and b equal to zero gives:

$$\boldsymbol{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\boldsymbol{x}_n)\,, \mathbf{0} = \sum_{n=1}^{N} a_n t_n$$

- Eliminating $\boldsymbol{w}$ and b from $L(\boldsymbol{w}, b, \boldsymbol{a})$ results in the dual representation:

$$\tilde{L}(\boldsymbol{a}) = \textstyle\sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n\, t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m),$$

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{\phi}(\boldsymbol{x}_n)^T\boldsymbol{\phi}(\boldsymbol{x}_m)$$

$$a_n \geq 0, n = 1, .. N, 0 = \sum_{n=1}^{N} a_n t_n$$

Dual Formulation

$$\max_{\boldsymbol{a}} \tilde{L}(\boldsymbol{a}),$$

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m),$$

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{\phi}(\boldsymbol{x}_n)^T \boldsymbol{\phi}(\boldsymbol{x}_m)$$

$$a_n \geq 0, n = 1, .. N, 0 = \sum_{n=1}^{N} a_n t_n$$

- The solution to a quadratic programming problem in M variables is $\mathcal{O}(M^3)$.
- The dual formulation involves maximizing over N variables. For $M \ll N$, the dual problem appears disadvantageous.
- However, the model is reformulated using kernels, and so the maximum margin classifier can be applied efficiently to feature spaces whose dimensionality exceeds N including infinite feature spaces.
- $k(\boldsymbol{x}, \boldsymbol{x}')$ being positive definite ensures that $\tilde{L}(\boldsymbol{a})$ is bounded above, thus leading to a well posed optimization.

Quadratic Programming – KKT Conditions

- In order to classify new data points using the trained model, we evaluate the sign of $y(\boldsymbol{x})$ defined by $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b$

- This can be expressed in terms of $\{a_n\}$ and the kernel function by substituting $\boldsymbol{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\boldsymbol{x}_n)$ to give

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n \, k(\boldsymbol{x}, \boldsymbol{x}_n) + \mathrm{b}$$

- Our constraint optimization problem satisfies the following Karush-Kuhn-Tucker (KKT) conditions:

$$a_n \geq 0$$

$$t_n y(\boldsymbol{x}_n) - 1 \geq 0$$

$$a_n(t_n y(\boldsymbol{x}_n) - 1) = 0$$

KKT Conditions

$$a_n \geq 0$$
$$t_n y(\boldsymbol{x}_n) - 1 \geq 0$$
$$a_n(t_n y(\boldsymbol{x}_n) - 1) = 0$$

- Thus for every $\boldsymbol{x}_n$, either $a_n = 0$ or $t_n y(\boldsymbol{x}_n) = 1$.
- Any data point for which $a_n = 0$ will not appear in $y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + \mathrm{b}$ and hence plays no role in making predictions for new data points.
- The remaining data points are called *support vectors*, and because they satisfy $t_n y(\boldsymbol{x}_n) = 1$, they correspond to points that lie on the maximum margin hyperplanes in feature space.
- Once the SVM model is trained, a significant proportion of the data points can be discarded and only the support vectors retained.

Computing b

- Having found the value of $\boldsymbol{a}$, we can determine b by noting that any support $\boldsymbol{x}_n$ satisfies $t_n y(\boldsymbol{x}_n) = 1$.

- Using $y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n \, k(\boldsymbol{x}, \boldsymbol{x}_n) + \mathrm{b}$ this gives:

$$t_n \left(\sum_{m \in \mathcal{S}} a_m t_m \, k(\boldsymbol{x}_n, \boldsymbol{x}_m) + \mathrm{b} \right) = 1$$

- Here $\mathcal{S}$ denotes the indices of the support vectors. A numerically stable solution is obtained by multiplying by t_n, use $t_n^2 = 1$, and average over all support vectors before solving for b to give

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m \, k(\boldsymbol{x}_n, \boldsymbol{x}_m) \right)$$

Error Function

- For comparison with alternative models, we can express the maximum margin classifier in terms of minimization of an error function, with a quadratic regularizer,

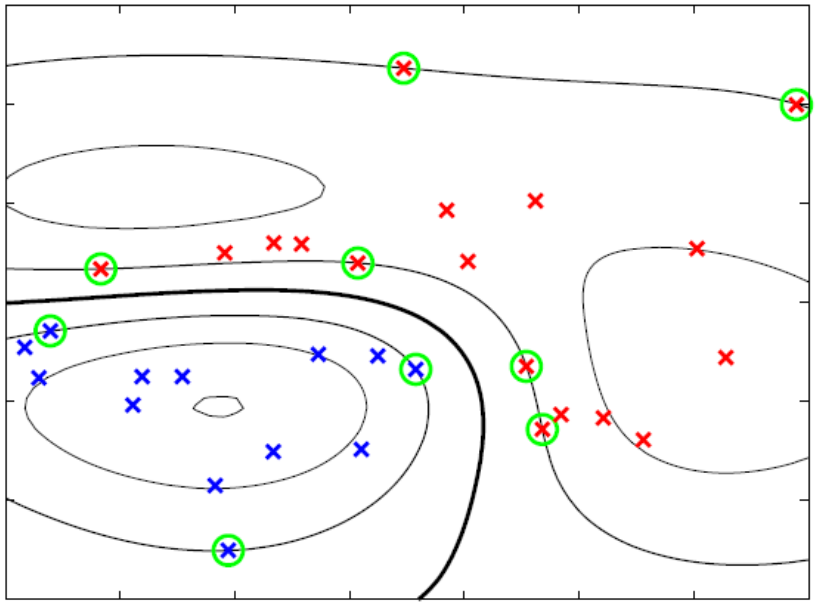
$$\sum_{n=1}^{N} E_\infty(t_n y(\boldsymbol{x}_n) - 1) + \lambda \|\boldsymbol{w}\|^2$$

- $E_\infty(z)$ is a function that is zero if $z \geq 0$ and ∞ otherwise and ensures that the constraints $t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b) \geq 1$ are satisfied.

- Note that as long as $\lambda > 0$, its precise value plays no role here.

Example: SVM with Gaussian Kernel

- Example of synthetic data from two classes in 2D showing contours of constant $y(\boldsymbol{x})$ obtained from a SVM having a Gaussian kernel function.

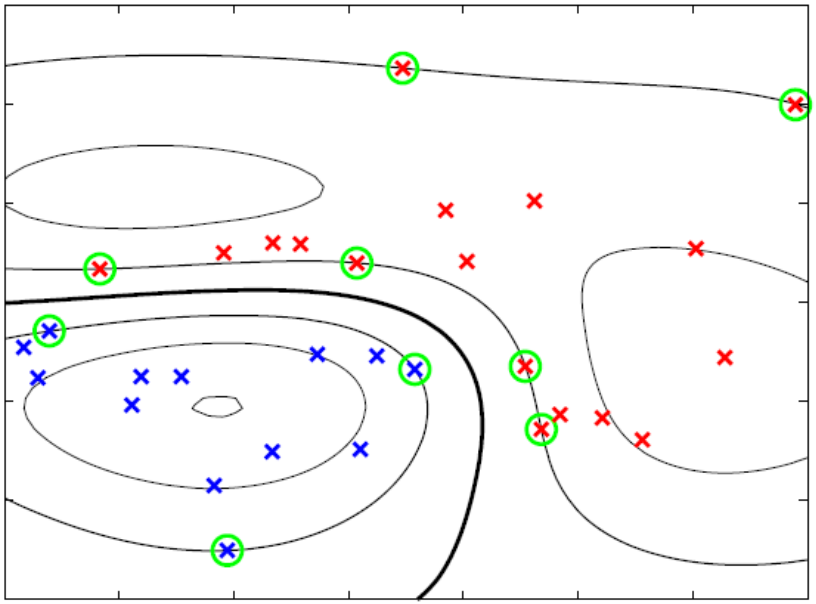


- Also shown the decision boundary, the margin boundaries, and the support vectors.

- Although the data set is not linearly separable in the 2D data space $\boldsymbol{x}$, it is linearly separable in the nonlinear feature space defined implicitly by the kernel function.

- Thus the training data points are perfectly separated in the original data space.

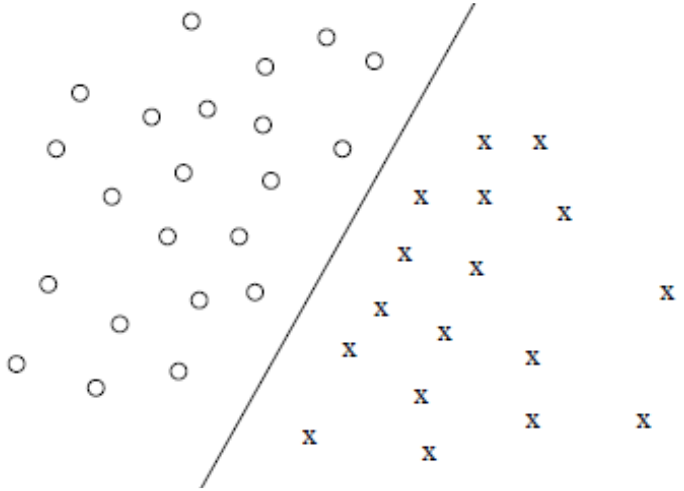
Example: SVM with Gaussian Kernel



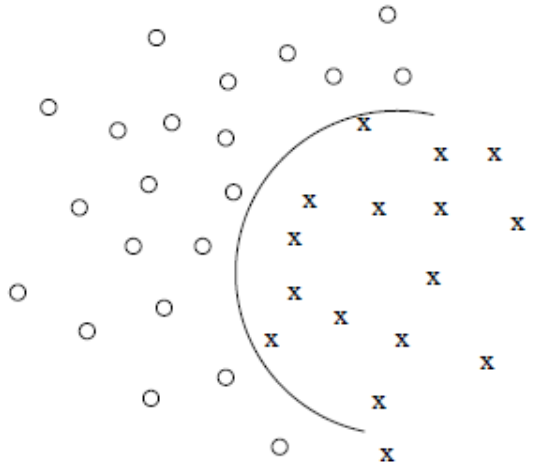
- The maximum margin hyperplane is defined by the location of the support vectors.
- Other data points can be moved around freely (so long as they remain outside the margin region) without changing the decision boundary, and so the solution remains independent of such data points.

Non-linear Classifier

- We easily obtain a non-linear classifier by using the kernel trick. Map $\boldsymbol{x} = [x_1\ x_2]$ into $\boldsymbol{\phi}(\boldsymbol{x}) = \left[x_1^2\ x_2^2\ \sqrt{2}x_1x_2\ \sqrt{2}x_1\ \sqrt{2}x_2\ 1\right]$



Linear separator in the feature ϕ-space



Non-linear separator in the original x-space

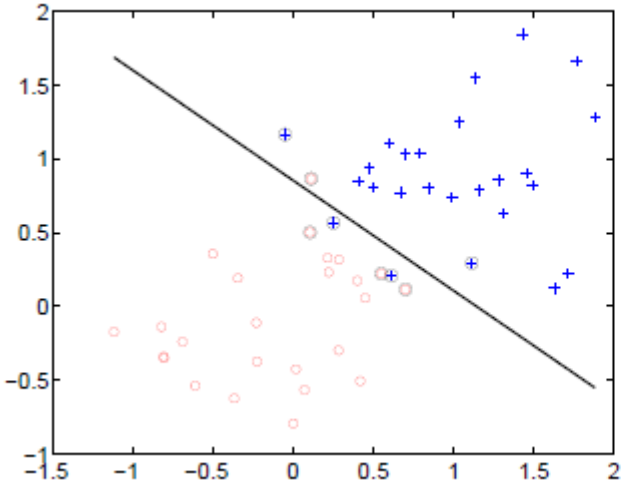
- The algorithm only needs dot products of the form:

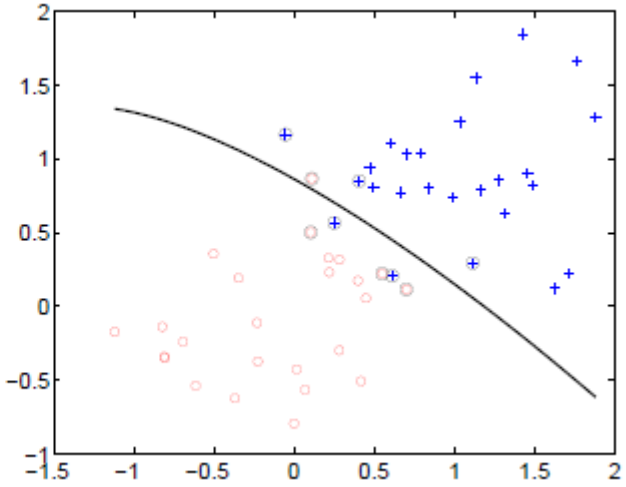
$$\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{x}') = x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_2x_1'x_2' + 2x_1\ x_1' + 2x_2\ x_2' + 1 = (1 + \boldsymbol{x}^T\boldsymbol{x}')^2$$

- So the inner products can be evaluated without ever explicitly constructing the feature vectors $\boldsymbol{\phi}(\boldsymbol{x})$!

SVM Classification: Examples

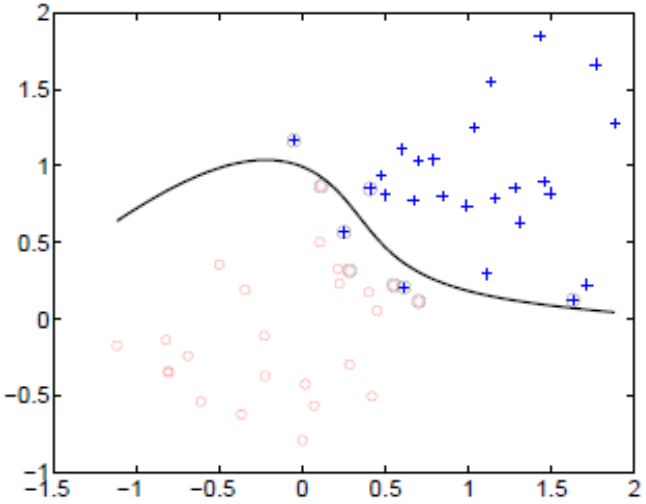
linear

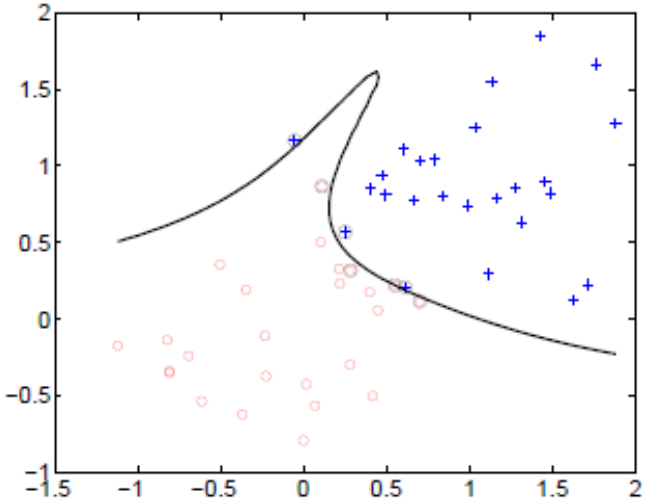




2nd order polynomial

4th order polynomial





8th order polynomial

Overlapping Class Distributions

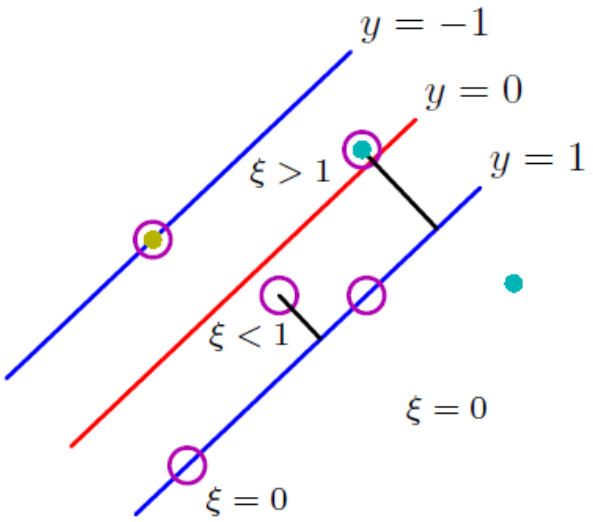
- Up to now we assumed that the training data points are linearly separable in the feature space $\boldsymbol{\phi}(\boldsymbol{x})$. The resulting SVM gives exact separation of the training data in the space $\boldsymbol{x}$, although the corresponding decision boundary will be nonlinear.
- In practice, the class-conditional distributions may overlap, and exact separation of the training data can lead to poor generalization.
- Need to allow some of the training points to be misclassified. From

$$\sum_{n=1}^{N} E_{\infty}(t_n y(\boldsymbol{x}_n) - 1) + \lambda \|\boldsymbol{w}\|^2$$

we see that in the case of separable classes, we implicitly used an error function that gave infinite error if a data point was misclassified and zero error if it was classified correctly, and then optimized the model parameters to maximize the margin.

Overlapping Class Distributions

- We now modify this approach so that data points are allowed to be on the 'wrong side' of the margin boundary, but with a penalty that increases with the distance from that boundary.
- It is convenient to make this penalty a linear function of this distance.
- We introduce *slack variables*, $\xi_n \geq 0, n = 1, \ldots, N$, with one slack variable for each training data point.
- $\xi_n = 0$ for data points that are on or inside the correct margin boundary
- $\xi_n = |t_n - y(\boldsymbol{x}_n)|$ for other points. Thus a data point that is on the decision boundary $y(\boldsymbol{x}_n) = 0$ will have $\xi_n = 1$, and points with $\xi_n > 1$ will be misclassified.



- Bennett, K. P. (1992). Robust linear programming discrimination of two linearly separable sets. *Optimization Methods and Software* **1**, 23–34.
- Cortes, C. and V. N. Vapnik (1995). Support vector networks. *Machine Learning* **20**, 273–297.

Overlapping Class Distributions

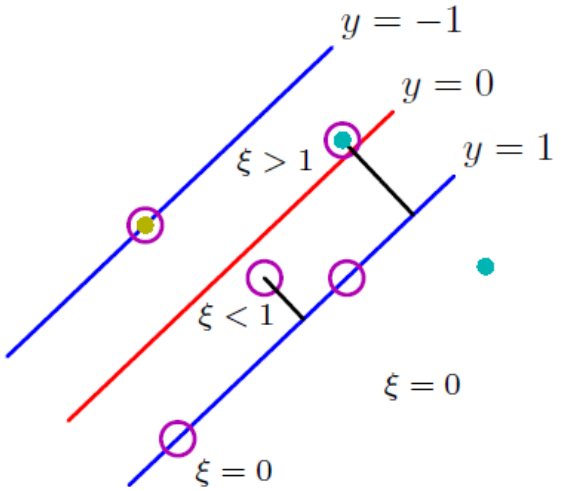
- The exact classification constraints

$$t_n(\boldsymbol{w}^T\phi(\boldsymbol{x}_n) + b) \geq 1$$

are then replaced with

$$t_n y(\boldsymbol{x}_n) \geq 1 - \xi_n, n = 1, \ldots, N$$

in which the slack variables are constrained to satisfy $\xi_n \geq 0$.



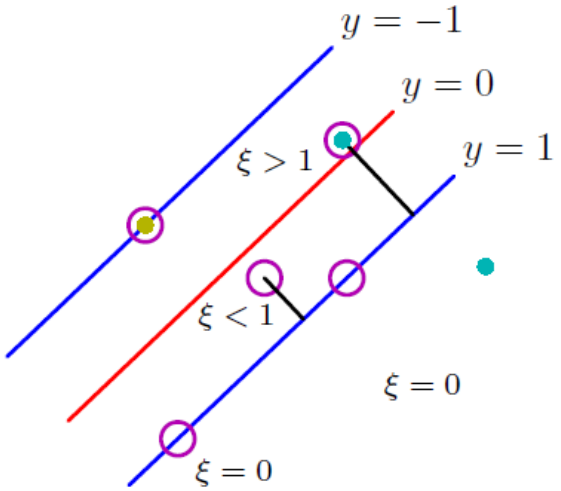
- Data points for which $\xi_n = 0$ are correctly classified and are either on the margin or on the correct side of the margin.

- Points for which $0 < \xi_n \leq 1$ lie inside the margin, but on the correct side of the decision boundary, and those data points for which $\xi_n > 1$ lie on the wrong side of the decision boundary and are misclassified.

- This is the *soft margin* constraint.

Overlapping Class Distributions

- This allows some of the training data points to be misclassified.



- While slack variables allow for overlapping class distributions, this framework is still sensitive to outliers because the penalty for misclassification increases linearly with ξ.

- Our goal is now to maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary. We minimize

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|\boldsymbol{w}\|^2$$

where $C > 0$ controls the trade-off between the slack variable penalty and the margin.

Overlapping Class Distributions

$$C\sum_{n=1}^{N}\xi_n+\frac{1}{2}\|\boldsymbol{w}\|^2$$

- Because any point that is misclassified has $\xi_n > 1$, it follows that $\sum_{n=1}^{N}\xi_n$ is an upper bound on the number of misclassified points.
- C is like the inverse of a regularization coefficient because it controls the trade-off between minimizing training errors and controlling model complexity.
- As $C \to \infty$, we will recover the earlier support vector machine for separable data.

Overlapping Class Distributions

- We now wish to minimize

$$C\sum_{n=1}^{N}\xi_n+\frac{1}{2}\|\boldsymbol{w}\|^2$$

subject to the constraints

$$t_n y(\boldsymbol{x}_n)\geq 1-\xi_n, n=1,\ldots,N$$

together with $\xi_n\geq 0$.

- The corresponding Lagrangian is given by

$$L(\boldsymbol{w},b,\boldsymbol{\xi},\boldsymbol{a},\boldsymbol{\mu})=\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{n=1}^{N}\xi_n-\sum_{n=1}^{N}a_n\{t_n y(\boldsymbol{x}_n)-1+\xi_n\}-\sum_{n=1}^{N}\mu_n\xi_n$$

where $a_n\geq 0, \mu_n\geq 0$ are Lagrange multipliers for the two constraints.

KKT Conditions

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\{t_n y(\boldsymbol{x}_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n$$

- The KKT conditions for $n = 1, \ldots, N$ are as follows:

$$a_n \geq 0$$

$$t_n y(\boldsymbol{x}_n) - 1 + \xi_n \geq 0$$

$$a_n\{t_n y(\boldsymbol{x}_n) - 1 + \xi_n\} = 0$$

$$\mu_n \geq 0$$

$$\xi_n \geq 0$$

$$\mu_n\xi_n = 0$$

Quadratic Programming

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\{t_n y(\boldsymbol{x}_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n$$

- Optimizing wrt $\boldsymbol{w}, b, \xi_n$ gives:

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\boldsymbol{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{n=1}^{N} a_n t_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \rightarrow a_n = C - \mu_n$$

- Use these results to eliminate $\boldsymbol{w}, b, \xi_n$ from $L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{a}, \boldsymbol{\mu})$.

Quadratic Programming

$$\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

- This is identical to the separable case but with different constraints.
- $a_n \geq 0$ is required as they are Lagrange multipliers.
- $a_n = C - \mu_n$ together with $\mu_n \geq 0$ implies $a_n \leq C$.
- We thus need to maximize $\tilde{L}(\boldsymbol{a})$ wrt a_n, $n = 1, .., N$ subject to

$$0 \leq a_n \leq C$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

- Predictions are again made using: $y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n \, k(\boldsymbol{x}, \boldsymbol{x}_n) + \mathrm{b}$

Interpretation of the Solution

- As before, a subset of the data points may have $a_n = 0$, in which case they do not contribute to the predictive model $y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + \mathrm{b}$
- The remaining data points constitute the support vectors. They are such that $a_n > 0$ and hence from $a_n\{t_n y(\boldsymbol{x}_n) - 1 + \xi_n\} = 0$ must satisfy:

$$t_n y(\boldsymbol{x}_n) = 1 - \xi_n$$

- If $a_n < C$ then $a_n = C - \mu_n$ implies $\mu_n > 0$ which from $\mu_n \xi_n = 0$ requires $\xi_n = 0$ and hence such points lie in the margin.
- Points with $a_n = C$ can lie inside the margin and can either be correctly classified if $\xi_n \leq 1$ or misclassified if $\xi_n > 1$.

Determining b

- To determine b in $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b$, we note that those support vectors for which $0 < a_n < C$ have $\xi_n = 0$ so that $t_n y(\boldsymbol{x}_n) = 1$ and hence will satisfy:

$$t_n\left(\sum_{m\in\mathcal{S}} a_m t_m\, k(\boldsymbol{x}_n, \boldsymbol{x}_m) + \mathrm{b}\right) = 1$$

- A numerically stable solution is obtained

$$b = \frac{1}{N_{\mathcal{M}}}\sum_{n\in\mathcal{M}}\left(t_n - \sum_{m\in\mathcal{S}} a_m t_m\, k(\boldsymbol{x}_n, \boldsymbol{x}_m)\right)$$

- Here $\mathcal{M}$ the sets of indices of data points having $0 < a_n < C$.

ν-SVM

- An alternative equivalent formulation of the SVM involves maximizing:

$$\tilde{L}(\boldsymbol{a}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

subject to the following constraints:

$$0 \leq a_n \leq {}^{1}/_{N}$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

$$\sum_{n=1}^{N} a_n \geq \nu$$

- Scholkopf, B., A. Smola, R. C.Williamson, and P. L. Bartlett (2000). New support vector algorithms. *Neural Computation* **12**(5), 1207–1245

ν-SVM

$$\tilde{L}(\boldsymbol{a}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

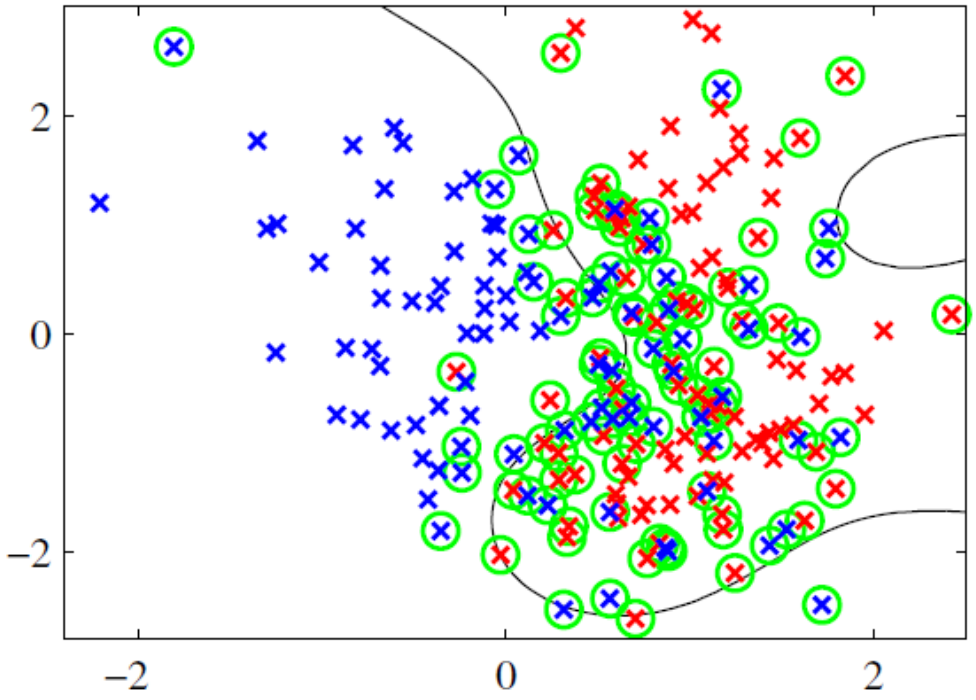
$$0 \leq a_n \leq {}^{1}/_{N}, \qquad \sum_{n=1}^{N} a_n t_n = 0, \qquad \sum_{n=1}^{N} a_n \geq \nu$$

- ν, which replaces C, can be interpreted as both
  - ✓ an upper bound on the fraction of *margin errors* (points for which $\xi_n > 0$ and hence which lie on the wrong side of the margin boundary and which may or may not be misclassified) and
  - ✓ a lower bound on the fraction of support vectors.

ν-SVM

- An example of the ν-SVM applied to a synthetic data set is shown below.

- Gaussian kernels $\exp(-\gamma||\boldsymbol{x} - \boldsymbol{x}'||^2)$ are used with $\gamma = 0.45$.



ν-SVM

- Although predictions for new inputs are made using only the support vectors, the training phase (i.e., the determination of the parameters $\boldsymbol{a}$ and b) makes use of the whole data set.

- Thus it is important to have efficient algorithms for the quadratic programming problem.

- $\tilde{L}(\boldsymbol{a})$ is quadratic and so any local optimum will also be a global optimum provided the constraints define a convex region (which they do as a consequence of being linear).

- Direct solution of the quadratic programming problem using traditional techniques is often infeasible due to computation and memory requirements.

Chunking

- *Chunking* exploits the fact that the value of the Lagrangian is unchanged if we remove the rows and columns of the kernel matrix corresponding to Lagrange multipliers that have zero value.

- This allows the full quadratic programming problem to be broken down into a series of smaller ones, whose goal is eventually to identify all of the nonzero Lagrange multipliers and discard the others.

- Chunking can be implemented using *protected conjugate gradients*.

- Although chunking reduces the size of the matrix in the quadratic function from the number of data points squared to approximately the number of nonzero Lagrange multipliers squared, even this may be too big to fit in memory for large-scale applications.

- Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining* **2**(2), 121–167.
- Vapnik, V. N. (1982). *Estimation of dependences based on empirical data*. Springer.

Decomposition Methods

- *Decomposition methods* also solve a series of smaller quadratic programming problems (each of these is of a fixed size), and so the technique can be applied to large data sets.
- One of the most popular approaches to training support vector machines is called *sequential minimal optimization*.
- It takes the concept of chunking to the extreme limit and considers just two Lagrange multipliers at a time. In this case, the subproblem can be solved analytically, avoiding quadratic programming.
- Heuristics are given for choosing the pair of Lagrange multipliers in each step. The scaling with the number of data points is between linear and quadratic depending on the application.

- Osuna, E., R. Freund, and F. Girosi (1996). Support vector machines: training and applications. A.I. Memo AIM-1602, MIT.
- Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Scholkopf, C. J. C. Burges, and A. J. Smola (Eds.), *Advances in Kernel Methods – Support Vector Learning*, pp. 185–208. MIT Press.

SVM and Curse of Dimensionality

- Kernel functions correspond to inner products in feature spaces that can have high, or even infinite, dimensionality.

- By working directly in terms of the kernel function, without introducing the feature space explicitly, does not mean that SMVs avoid the curse of dimensionality.

- There are constraints amongst the feature values that restrict the effective dimensionality of feature space. To see this consider a 2nd order polynomial kernel that we can expand in terms of its components

$$k(\boldsymbol{x},\boldsymbol{z}) = (1 + \boldsymbol{x}^T\boldsymbol{z})^2 = (1 + x_1z_1 + x_2z_2)^2$$
$$= 1 + 2x_1z_1 + 2x_2z_2 + x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2$$
$$= \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2\right)\left(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1z_2, z_2^2\right)^T$$
$$= \boldsymbol{\phi}(\boldsymbol{x})^T\,\boldsymbol{\phi}(\boldsymbol{z})$$

SVM and Curse of Dimensionality

- This kernel function therefore represents an inner product in a feature space having six dimensions, in which the mapping from input space to feature space is described by the vector function $\boldsymbol{\phi}(\boldsymbol{x})$.

- However, the coefficients weighting these different features are constrained to have specific forms.

- Thus any set of points in the original 2D space $\boldsymbol{x}$ would be constrained to lie exactly on a 2D nonlinear manifold embedded in the six-dimensional feature space.

Probabilistic Predictions with SVM

- The SVM does not provide probabilistic outputs but makes classification decisions for new input vectors.
- Modifications to the SVM were introduced to allow the trade-off between false positive and false negative errors to be controlled.
- To address probabilistic predictions of the class label t for new inputs $\boldsymbol{x}$, Platt (2000) has proposed fitting a logistic sigmoid to the outputs of a previously trained SVM. The required conditional probability is

$$p(t = 1|\boldsymbol{x}) = \sigma\left(Ay(\boldsymbol{x}) + B\right)$$

where $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b$.

- Veropoulos, K., C. Campbell, and N. Cristianini (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI99), Workshop ML3*, pp. 55–60.
- Platt, J. C., N. Cristianini, and J. Shawe-Taylor (2000). Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K. R. Muller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12, pp. 547–553. MIT Press.
- Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *J. of Machine Learning Research 1*, 211–244.

Probabilistic Predictions with SVM

$$p(t = 1|\boldsymbol{x}) = \sigma\left(Ay(\boldsymbol{x}) + B\right)$$

where $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b$.

- Values for A and B are found by minimizing the cross-entropy error function defined by a training set consisting of pairs of $y(\boldsymbol{x}_n)$ and t_n.
- The data used to fit the sigmoid needs to be independent of that used to train the original SVM in order to avoid severe over-fitting. This two-stage approach is equivalent to assuming that the output $y(\boldsymbol{x})$ of the SVM represents the log-odds of $\boldsymbol{x}$ belonging to class $t = 1$.
- The SVM can give a poor approximation to the posterior probabilities.

- Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *J. of Machine Learning Research 1*, 211–244.

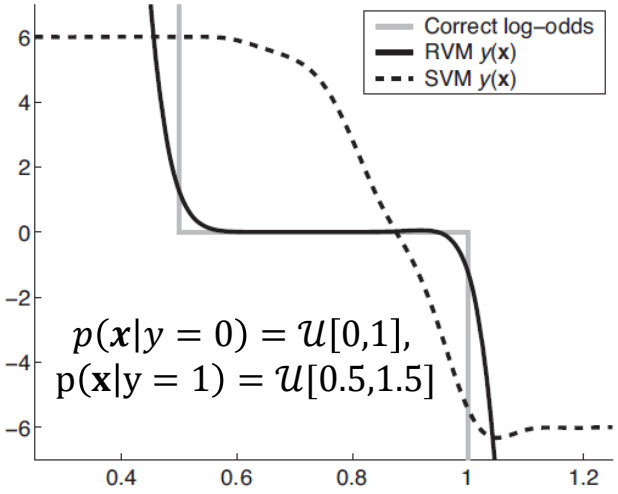
Probabilistic Response

- An SVM classifier produces a hard-labeling, $\hat{y}(\boldsymbol{x}) = sign(f(\boldsymbol{x}))$.

- A heuristic approach to producing confidence in our prediction is to interpret $f(\boldsymbol{x})$ as the log-odds ratio, $\log \frac{p(y=1|\boldsymbol{x})}{p(y=0|\boldsymbol{x})}$ and convert the output of an SVM to a probability using

$$p(y = 1|\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(af(\boldsymbol{x}) + b)$$

where a, b can be estimated by maximum likelihood on a separate validation set (using the training set to estimate a and b leads to severe overfitting.)



- The log-odds of class 1 over class 0 should be zero in [0.5, 1.0], and infinite outside this region.
- We sampled 1000 points from the model, and then fit an RVM and an SVM with a Gaussian kenel of width 0.1.
- Both models can perfectly capture the decision boundary, and achieve a generalizaton error of 25%, which is Bayes optimal in this problem.
- The probabilistic output from the RVM is a good approximation to the true log-odds, but this is not the case for the SVM.

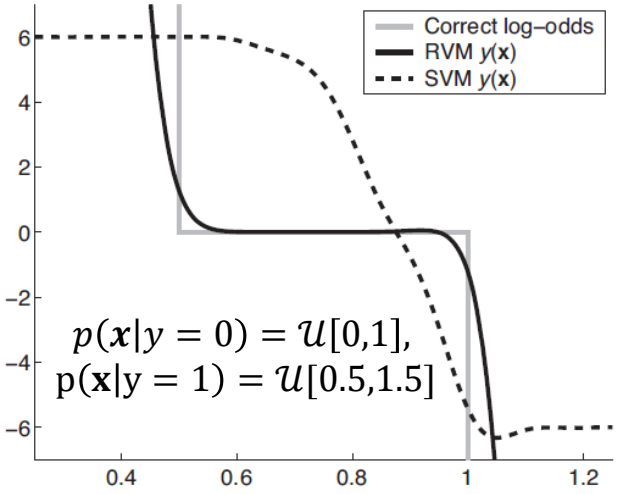
Probabilistic Response

- An SVM classifier produces a hard-labeling, $\hat{y}(\boldsymbol{x}) = sign(f(\boldsymbol{x}))$.

- A heuristic approach to producing confidence in our prediction is to interpret $f(\boldsymbol{x})$ as the log-odds ratio, $\log \frac{p(y=1|\boldsymbol{x})}{p(y=0|\boldsymbol{x})}$ and convert the output of an SVM to a probability using

$$p(y = 1|\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(af(\boldsymbol{x}) + b)$$

where a, b can be estimated by maximum likelihood on a separate validation set (using the training set to estimate a and b leads to severe overfitting.)



- Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *J. of Machine Learning Research 1*, 211–244.

Relation to Logistic Regression

- We can re-cast the SVM for nonseparable distributions in terms of the minimization of a regularized error function.

- This will also allow us to highlight similarities/differences from the logistic regression.

- As a reminder for data points that are on the correct side of the margin boundary, and which therefore satisfy $y_n t_n \geq 1$, we have $\xi_n = 0$, and for the remaining points we have $\xi_n = 1 - y_n t_n$.

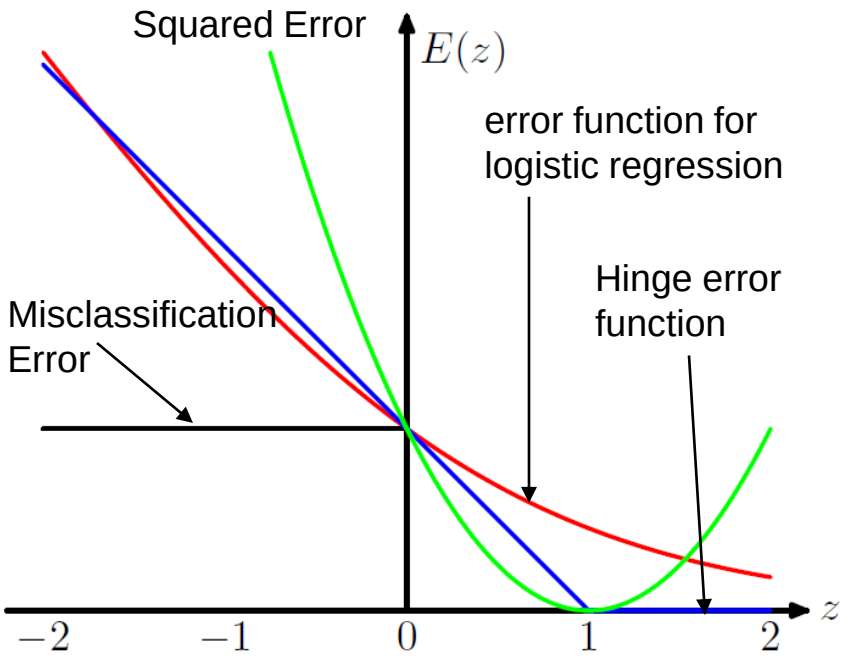
- Thus the objective function $C \sum_{n=1}^{N} \xi_n + \frac{1}{2}\|\boldsymbol{w}\|^2$ can be written (up to a multiplicative constant) in the form

$$\sum_{n=1}^{N} E_{SV}(y_n t_n) + \lambda \|\boldsymbol{w}\|^2$$

where $\lambda = (2C)^{-1}$, and $E_{SV}(\cdot)$ is the *hinge* error function defined by $E_{SV}(y_n t_n) = [1 - y_n t_n]_+$ where $[\quad]_+$ denotes the positive part.

Relation to Logistic Regression

- Plot of the 'hinge' error function in SVM, along with the error function for logistic regression, rescaled by a factor of $1/\ln(2)$ so that it passes through the point (0, 1).



- When we considered the logistic regression model we found it convenient to work with target variable $t \in \{0, 1\}$.

  For comparison with the SVM, we first reformulate maximum likelihood logistic regression using the target variable $t \in \{-1, 1\}$.

- To do this, we note that $p(t = 1|y) = \sigma(y)$ where $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b$ and $\sigma(y)$ is the logistic sigmoid function.

Error Function for Logistic Regression

- It follows that $p(t = -1|y) = 1 - \sigma(y) = \sigma(-y)$, where we have used the properties of the logistic sigmoid function, and so we can write $p(t|y) = \sigma(yt)$.

- We can construct an error function by taking the negative logarithm of the likelihood function that with a quadratic regularizer takes the form

$$\sum_{n=1}^{N} E_{LR}(y_n t_n) + \lambda \|\boldsymbol{w}\|^2, \; E_{LR}(y_n t_n) = \ln(1 + \exp(-yt))$$

- For comparison with other error functions, we can divide by $\ln(2)$ so that the error function passes through the point (0, 1). This rescaled error function is also plotted in the earlier Fig. and has a similar form to the support vector error function.

- The key difference is that the flat region in $E_{SV}(y_n t_n)$ leads to sparse solutions. Both the logistic error and the hinge loss can be viewed as continuous approximations to the misclassification error.

Hinge Loss

- We have seen that the negative log-likelihood for the logistic regression model is

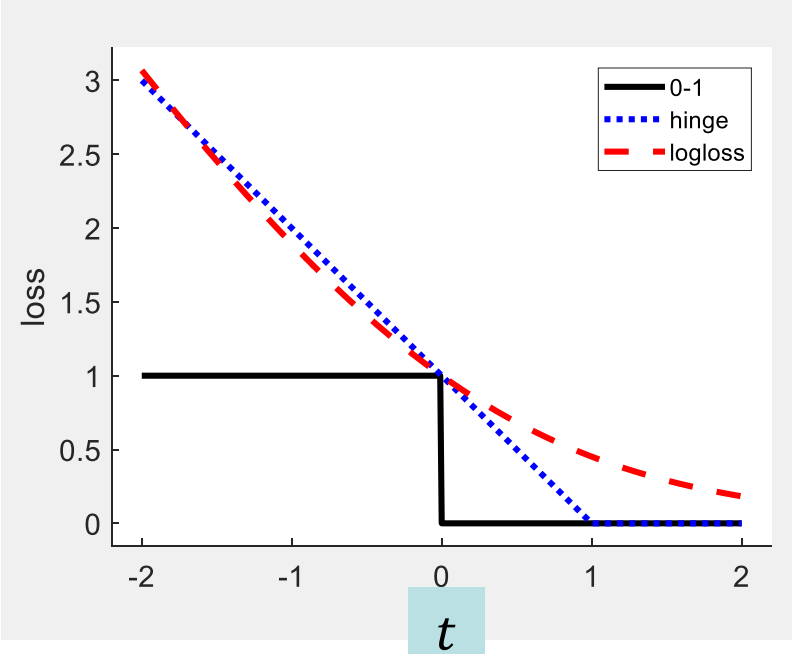
$$L_{nll}(y,t) = -logp(t|\boldsymbol{x},\boldsymbol{w}) = \log(1+e^{-yt})$$

- It is a convex upper bound on the $0-1$ risk of a binary classifier, where $y = f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b$ is the log odds ratio, and we have assumed the labels are $t \in \{1,-1\}$ rather than $\{0,1\}$.

- We can replace the NLL loss with the **hinge loss**

$$L_{hinge}(y,t) = \max(0,1-yt) = (1-yt)_+$$

The horizontal axis is the margin yt, the vertical axis is the loss. The log loss uses log base 2.



hingeLossPlot.m
from PMTK3

Hinge Loss

- Our optimization problem is as follows:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}(1 - t_i f(\boldsymbol{x}_i))_+$$

- Introducing slack variables we can write this as a quadratic program in $N + D + 1$ variables:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i, \qquad \xi_i \geq 0, t_i(\boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{w} + b) \geq 1 - \xi_i, i = 1,..N$$

- We can eliminate $\boldsymbol{w}$, b and ξ_i, and just solve the N dual variables, which correspond to the Lagrange multipliers for the constraints. Standard solvers take $\mathcal{O}(N^3)$ time. **Sequential minimal optimization** takes $\mathcal{O}(N^2)$. Linear SVMs, take $\mathcal{O}(N)$ time to train.

- Platt, J. (1998). Using analytic QP and sparseness to speed training of support vector machines. In *NIPS*.
- Joachims, T. (2006). Training Linear SVMs in Linear Time. In *Proc. Of the Int'l Conf. on Knowledge Discovery and Data Mining*.
- Bottou, L., O. Chapelle, D. DeCoste, and J. Weston (Eds.) (2007). *Large Scale Kernel Machines*. MIT Press.

SVMs for Classification

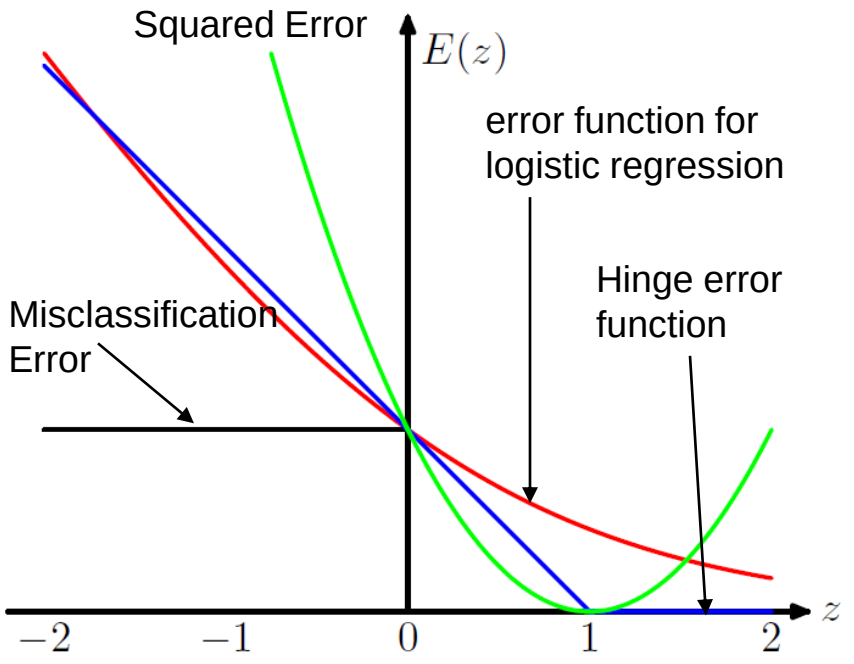
- We have shown that: $\widehat{\boldsymbol{w}} = \sum_{i=1}^{N} \alpha_i t_i \boldsymbol{\phi}(\boldsymbol{x}_i)\,, \alpha\ is\ sparse$
- The $\boldsymbol{x}_i$'s for which $\alpha_i > 0$ are called support vectors – these are points which are incorrectly classified or are classified correctly but are on or inside the margin.
- Making predictions:

$$\hat{y}(\boldsymbol{x}) = sgn(b + \widehat{\boldsymbol{w}}^T \boldsymbol{\phi}(\boldsymbol{x})) = sgn(b + \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}_i\ ,\ \boldsymbol{x}))$$

- It takes take $\mathcal{O}(sD)$ time to compute where $s \leq N$ is the number of support vectors.

Error Functions

- Another continuous error function that has sometimes been used to solve classification problems is the squared error.
- It has the property, however, of placing increasing emphasis on data points that are correctly classified but that are a long way from the decision boundary on the correct side.
- Such points will be strongly weighted at the expense of misclassified points, and so if the objective is to minimize the misclassification rate, then a monotonically decreasing error function would be a better choice.



Probabilistic Interpretation of SVM

- Thus the minimization objective for the Hinge loss function

$$C\sum_{i=1}^{N}(1-t_i[b+\boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{w}])^+ + \frac{1}{2}\|\boldsymbol{w}\|^2$$

  is the same as that based on logistic regression.

- The corresponding regularized loss takes the form:

$$\frac{1}{N}\sum_{i=1}^{N}(1-t_i[b+\boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{w}])^+ + \frac{1}{NC}\|\boldsymbol{w}\|^2,$$

- Here the regularization parameter is $\lambda = \frac{1}{NC}$

- The first term above corresponds to the negative log likelihood:

$$\frac{1}{N}\sum_{i=1}^{N} -logg(t_i[b+\boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{w}]) \quad , g(z) = \frac{1}{1+e^{-z}} = logistic\ function$$

Probabilistic Interpretation of SVM

- The corresponding regularized loss takes the form:

$$\frac{1}{N}\sum_{i=1}^{N}(1 - t_i[b + \boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{w}])^+ + \frac{1}{NC}\|\boldsymbol{w}\|^2,$$

- One in the pursue of a probabilistic interpretation of SVM can show:

$$exp(-2(1 - t_i[b + \boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{w}])^+)$$
$$= \int_0^\infty \frac{1}{\sqrt{2\pi\lambda_i}} exp\left(-\frac{1}{2}\frac{(1 + \lambda_i - t_i[w_0 + \boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{w}])^2}{\lambda_i}\right) d\lambda_i$$

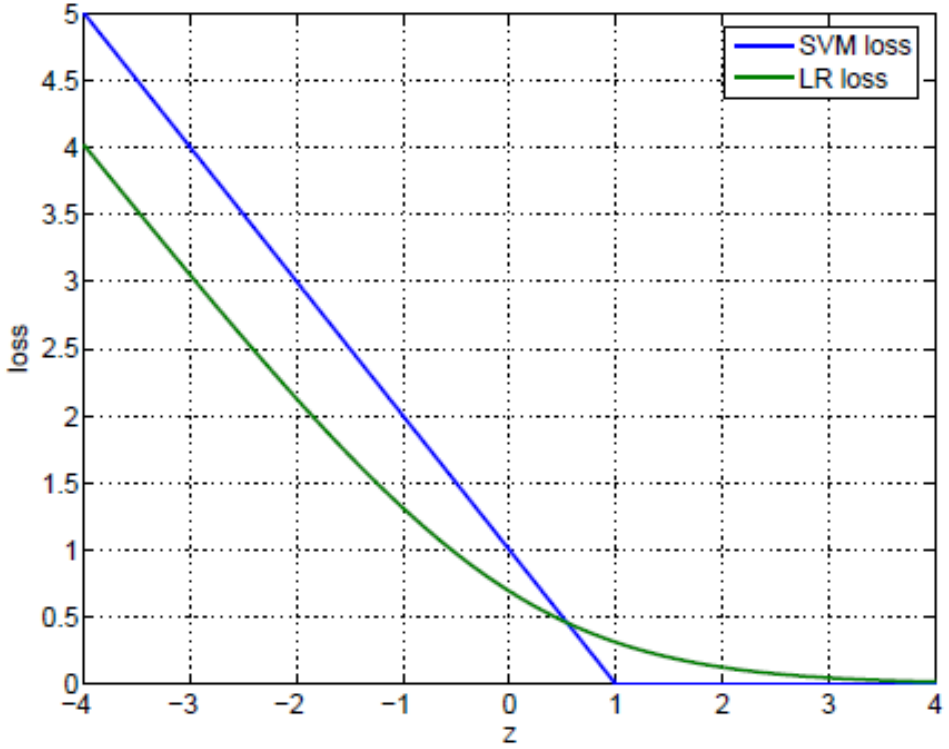
- This allows to fit the SVM with EM using λ_i as latent variables.

- Sollich, P. (2002). Bayesian methods for support vector machines: evidence and predictive class probabilities. *Machine Learning 46*, 21– 52.
- Polson, N. and S. Scott (2011). Data augmentation for support vector machines. *Bayesian Analysis 6*(1), 1–124.
- Franc, V., A. Zien, and B. Schoelkopf (2011). Support vector machines as probabilistic models. In *Intl. Conf. on Machine Learning*.

Probabilistic Interpretation of SVM

- Common objective the minimization of:

$$\frac{1}{N}\sum_{i=1}^{N} Loss(t_i[b + \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{w}]) + \frac{1}{NC}\|\boldsymbol{w}\|^2,$$

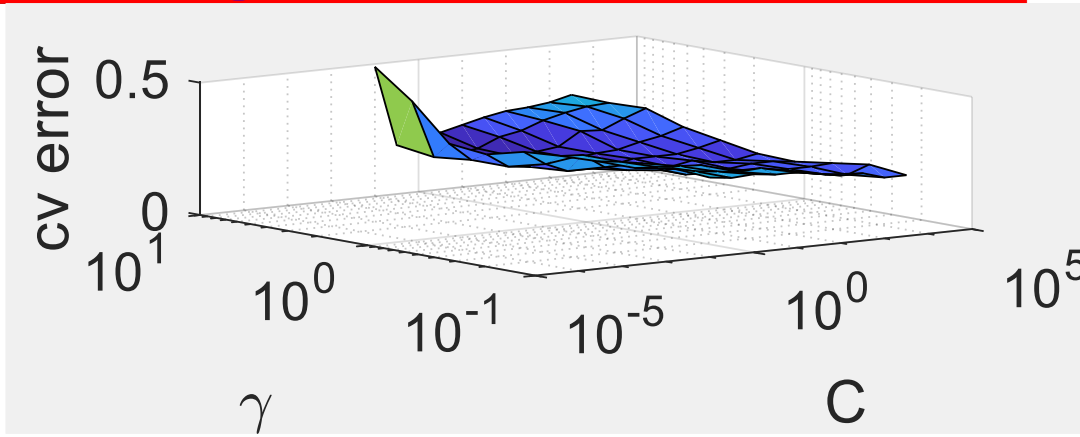


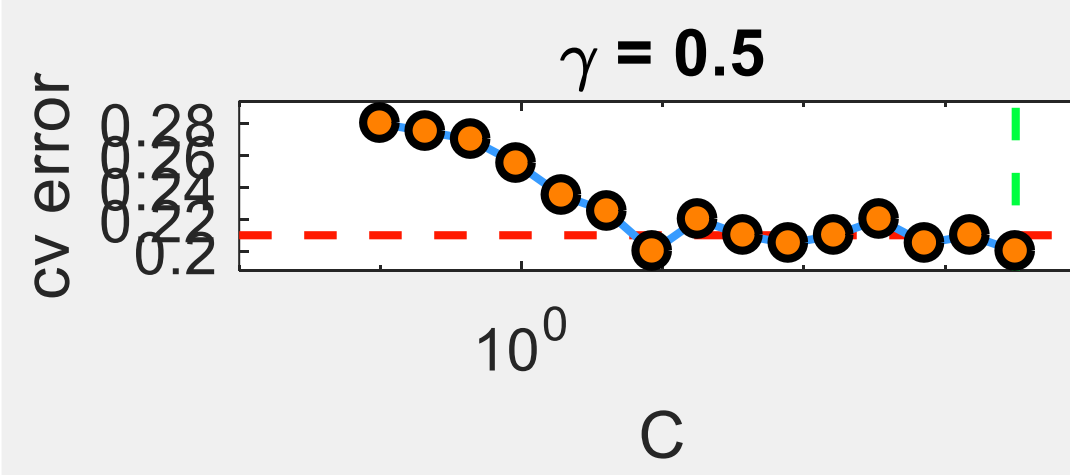
SVM Loss: $Loss(z) = (1 - z)^+$

LR Loss: $Loss(z) = \frac{1}{1+e^{-z}}$

Choosing C

- Need to specify the kernel parameters and C. Typically C is chosen by cross-validation.
- C interacts quite strongly with the kernel parameters.
- Consider an RBF kernel with precision $\gamma = \frac{1}{(2\sigma)^2}$
- If $\gamma = 5$, corresponding to narrow kernels, we need heavy regularization, and hence small C (so $\lambda = 1/C$ is big).
- If $\gamma = 1$, a larger value of C should be used.





CV estimate of the 0-1 risk as a function of C and γ.

svmCGammaDemo.m from PMTK3

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\boldsymbol{w}\|^2$$

Choosing C

- libsvm recommends using CV over a 2d grid with values $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$.

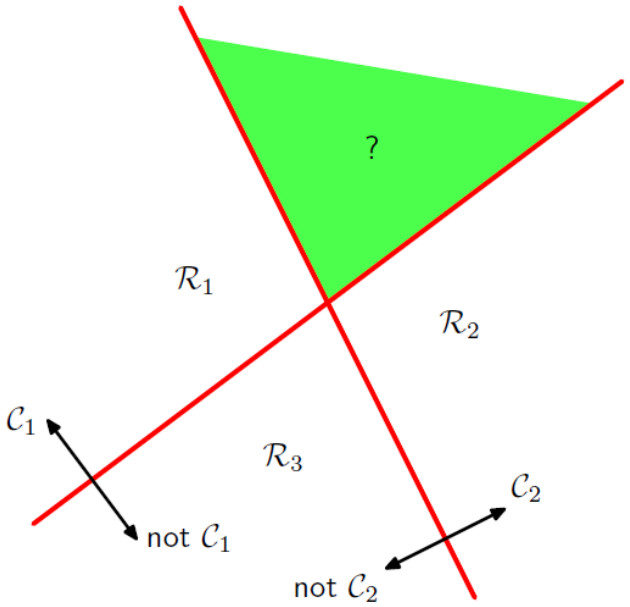
- Standardize the data first, for a spherical Gaussian kernel to make sense.

- To choose C efficiently, one can develop a path following the spirit of lars.
  - The basic idea is to start with λ large, so that the margin $1/||\boldsymbol{w}(\lambda)||$ is wide, and hence all points are inside of it and have $\alpha_i = 1$.
  - By slowly decreasing λ, a small set of points will move from inside the margin to outside, and their α_i values will change from 1 to 0, as they cease to be support vectors.
  - When λ is maximal, the function is completely smoothed, and no support vectors remain.

- Hastie, T., S. Rosset, R. Tibshirani, and J. Zhu (2004). The entire regularization path for the support vector machine. *J. of Machine Learning Research 5*, 1391–1415.

Multiclass SVMs

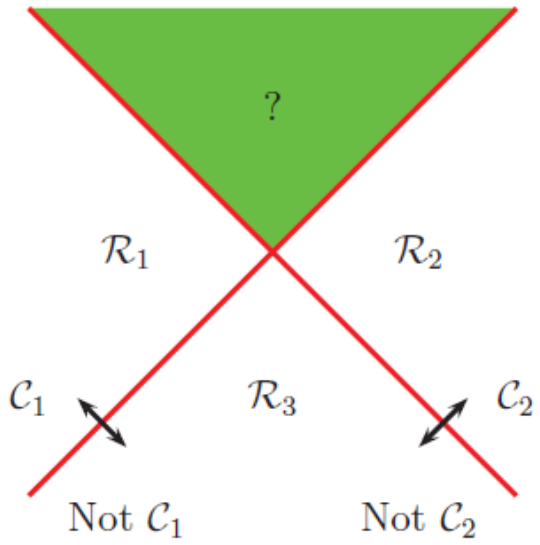
- The SVM is a two-class classifier. However, we often have to tackle $K > 2$ classes.
- Combine two-class SVMs in order to build a multiclass classifier.
- One approach (Vapnik, 1998) is to construct K separate SVMs, in which the k –th model $y_k(\boldsymbol{x})$ is trained using the data from class $\mathcal{C}_k$ as the positive examples and the data from the remaining $K - 1$ classes as the negative examples.
- This is known as the *one-versus-the-rest* approach.
- Using the decisions of the individual classifiers can lead to inconsistent results in which an input is assigned to multiple classes simultaneously.



- Vapnik, V. N. (1998). *Statistical learning theory*.Wiley.

Multiclass Classification

- The obvious approach is to use a **one-vs-all** approach. We train C binary classifiers, $f_c(x)$, where the data from class c is treated as positive, and the data from all the other classes as negative.
- This can result in ambiguously labeled regions.
- Can pick $\hat{y}(\boldsymbol{x}) = \underset{c}{\operatorname{argmax}} f_c(\boldsymbol{x})$. This will not work since the f_c functions don't have comparable magnitudes.



The one-versus-rest approach. The green region is predicted to be both class 1 and class 2.

- Weston, J. and C. Watkins (1999). Multi-class support vector machines. In *ESANN*.

Multiclass SVMs

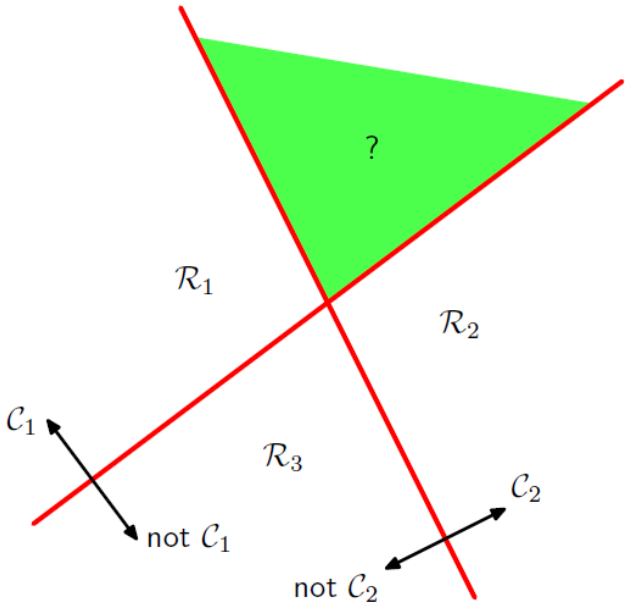
- This problem is sometimes addressed by making predictions for new inputs $\boldsymbol{x}$ using

$$y(\boldsymbol{x}) = \max_k y_k(\boldsymbol{x})$$

- This heuristic approach suffers from the problem that the different classifiers were trained on different tasks, and there is no guarantee that the $y_k(\boldsymbol{x})$ for different classifiers will have appropriate scales.

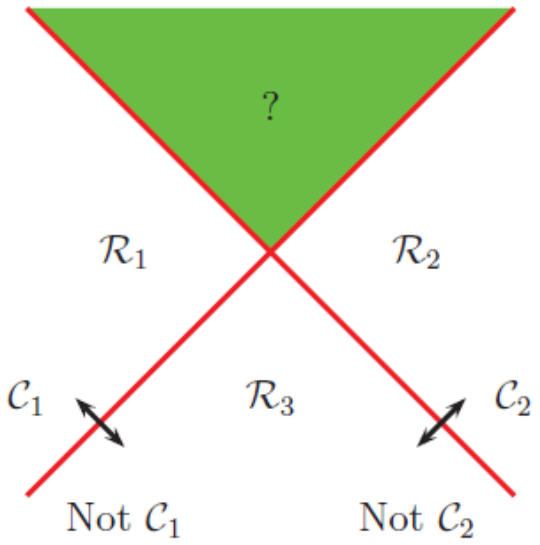
- Another problem with the one-versus-the-rest approach is that the training sets are imbalanced.

- If e.g. we have 10 classes with equal numbers of training data points, then the individual classifiers are trained on data sets comprising 90% negative examples and 10% positive examples, and the symmetry of the original problem is lost.



Multiclass Classification

- Each binary subproblem is likely to suffer from the **class imbalance** problem.
- E.g. consider 10 equally represented classes. When training f_1, we will have 10% positive examples and 90% negative examples, which can hurt performance.
- One can train all K classifiers simultaneousy, but the resulting method takes $\mathcal{O}(K^2N^2)$ time, instead of $\mathcal{O}(KN^2)$.



The one-versus-rest approach. The green region is predicted to be both class 1 and class 2.

- Weston, J. and C. Watkins (1999). Multi-class support vector machines. In *ESANN*.

Multiclass SVMs

- A variant of the one-versus-the-rest scheme was proposed by Lee *et al.* (2001) who modify the target values so that the positive class has target +1 and the negative class has target $-1/(K-1)$.

- Weston and Watkins (1999) define a single objective function for training all K SVMs simultaneously, based on maximizing the margin from each to remaining classes.

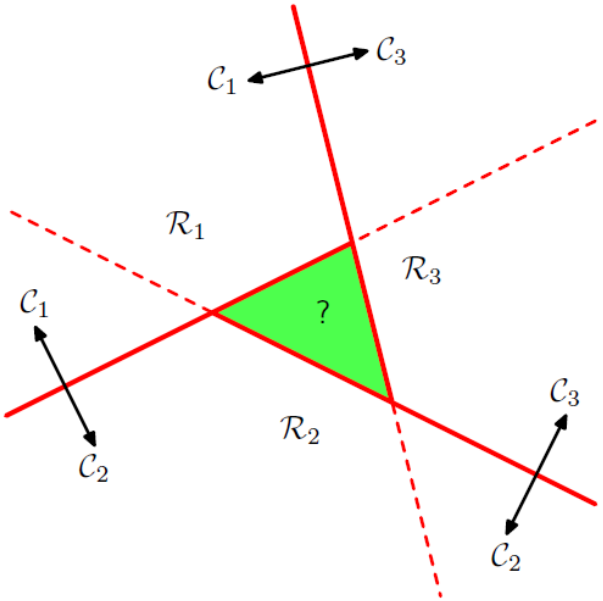
  However, this can result in much slower training because, instead of solving K separate optimization problems each over N data points with an overall cost of $\mathcal{O}(KN^2)$, a single optimization problem of size $(K-1)N$ must be solved giving an overall cost of $\mathcal{O}(K^2N^2)$

- Lee, Y., Y. Lin, and G.Wahba (2001). Multicategory support vector machines. Technical Report 1040, Department of Statistics, University of Madison, Wisconsin.
- Weston, J. and C. Watkins (1999). Multi-class support vector machines. In M. Verlysen (Ed.), *Proceedings ESANN'99, Brussels*. D-Facto Publications.

Multiclass SVMs

- One can train $K(K-1)/2$ different 2-class SVMs on all possible pairs of classes, and then to classify test points according to which class has the highest number of 'votes', *one-versus-one*.

- This can lead to ambiguities, for large K requires significantly more training time than the one-versus-the-rest approach and more computation is needed for predictions.

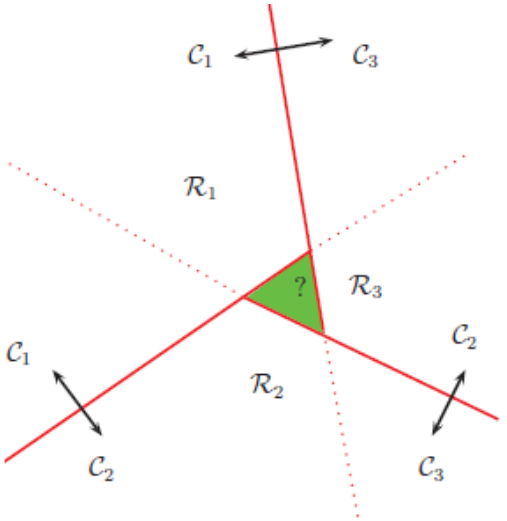


- One can organize the pairwise classifiers into a directed acyclic graph leading to the *DAGSVM*. For K classes, the DAGSVM has a total of $K(K-1)/2$ classifiers, and to classify a new test point only $K-1$ pairwise classifiers need to be evaluated, with the particular classifiers used depending on which path through the graph is traversed.

- Platt, J. C., N. Cristianini, and J. Shawe-Taylor (2000). Large margin DAGs for multiclass classification. In S. A. Solla, T. K. Leen, and K. R. Muller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12, pp. 547–553. MIT Press.

Multiclass Classification

- Another approach is the **one-versus-one.**
- We train $K(K-1)/2$ classifiers to discriminate all pairs $f_{c,c'}$. We then classify a point into the class which has the highest number of votes.
- This can also result in ambiguities.
- It takes $\mathcal{O}(K^2N^2)$ time to train and $\mathcal{O}(K^2N_{sv})$ to test each data point, where N_{sv} is the number of support vectors.
- Fundamental limitation of SVMs: they do not model uncertainty, so their output scores are not comparable across classes.



The one-versus-one approach.

- Allwein, E., R. Schapire, and Y. Singer (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *J. of Machine Learning Research*, 113–141.

Multiclass SVMs

- A different approach to multiclass classification, based on error-correcting output codes, was developed by Dietterich and Bakiri and applied to SVMs by Allwein *et al*.
- This can be viewed as a generalization of the voting scheme of the one-versus-one approach in which more general partitions of the classes are used to train the individual classifiers. The *K* classes themselves are represented as particular sets of responses from the two-class classifiers chosen, and together with a suitable decoding scheme, this gives robustness to errors and to ambiguity in the outputs of the individual classifiers.
- The one-versus-the-rest approach is the most widely used in spite of its ad-hoc formulation and its practical limitations.

- Dietterich, T. G. and G. Bakiri (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* **2**, 263–286.Scholkopf, B., J. Platt, J. Shawe-Taylor,
- Allwein, E. L., R. E. Schapire, and Y. Singer (2000). Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research* **1**, 113–141.

SVMs for Unsupervised Learning

- There are also *single-class SVMs*, which solve an unsupervised learning problem related to probability density estimation.

- Instead of modelling the density of data, these methods find a smooth boundary enclosing a region of high density. The boundary is chosen to represent a quantile of the density i.e. the probability that a data point drawn from the distribution will land inside that region is given by a number between 0 and 1 that is specified in advance.

- This is a more restricted problem than estimating the full density. Two approaches to this problem using SVMs have been proposed.

- Tax, D. and R. Duin (1999). Data domain description by support vectors. In M. Verleysen (Ed.), *Proceedings European Symposium on Artificial Neural Networks, ESANN*, pp. 251–256. D. Facto Press.
- Scholkopf, B., J. Platt, J. Shawe-Taylor, A. Smola, and R. C.Williamson (2001). Estimating the support of a high-dimensional distribution. *Neural Computation* **13**(7), 1433–1471.

Multiclass SVMs

- Scholkopf *et al.* (2001) tries to find a hyperplane that separates all but a fixed fraction *v* of the training data from the origin while at the same time maximizing the distance (margin) of the hyperplane from the origin.

- Tax and Duin (1999) look for the smallest sphere in feature space that contains all but a fraction *v* of the data points.

- For kernels $k(\boldsymbol{x}, \boldsymbol{x}')$ that are functions only of $\boldsymbol{x} - \boldsymbol{x}'$, the two algorithms are equivalent.

- Tax, D. and R. Duin (1999). Data domain description by support vectors. In M. Verleysen (Ed.), *Proceedings European Symposium on Artificial Neural Networks, ESANN*, pp. 251–256. D. Facto Press.
- Scholkopf, B., J. Platt, J. Shawe-Taylor, A. Smola, and R. C.Williamson (2001). Estimating the support of a high-dimensional distribution. *Neural Computation* **13**(7), 1433–1471.

SVMs for Regression

- We now extend support vector machines to regression problems.

- In simple linear regression, we minimize a regularized error function given by

$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

- To obtain sparse solutions, the quadratic error function is replaced by an ε–*insensitive error function* (Vapnik, 1995), which gives zero error if the absolute difference between the prediction $y(\boldsymbol{x})$ and the target t is less than ϵ where $\epsilon > 0$.

- A simple example of an -insensitive error function, having a linear cost associated with errors outside the insensitive region, is given by

$$\mathbb{E}_\epsilon\,(y(\mathbf{x}) - \mathrm{t}) = \begin{cases} 0, & \text{if } |y(x) - t| < \epsilon: \\ |y(\mathrm{x}) - t| - \epsilon, & \text{otherwise} \end{cases}$$

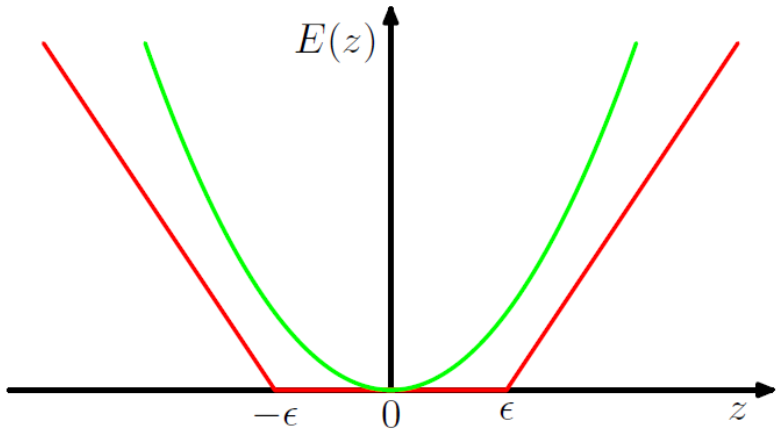
- Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer.

SVMs for Regression

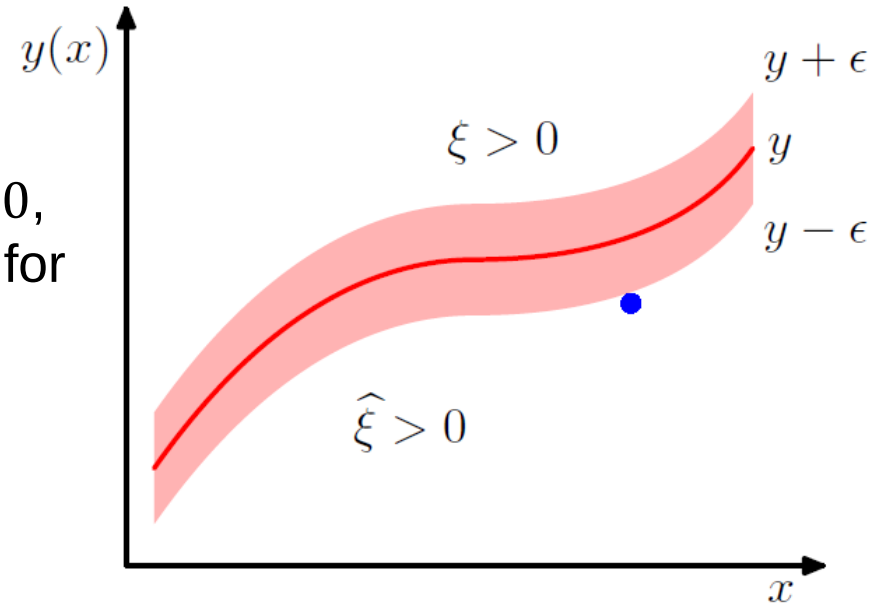
- We therefore minimize a regularized error function given by

$$C\sum_{n=1}^{N} E_\epsilon\left(y(\mathrm{x}_n) - t_n\right) + \frac{1}{2}\|\boldsymbol{w}\|^2$$

where $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b$.



- The (inverse) regularization parameter, C, appears in front of the error term.
- For each data point $\boldsymbol{x}_n$, we now need two slack variables $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$, where $\xi_n \geq 0$ corresponds to a point for which $t_n > y(\boldsymbol{x}_n) + \varepsilon$, and $\hat{\xi}_n > 0$, corresponds to a point for which $t_n < y(\boldsymbol{x}_n) - \varepsilon$.



- Schoelkopf, B. and A. Smola (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

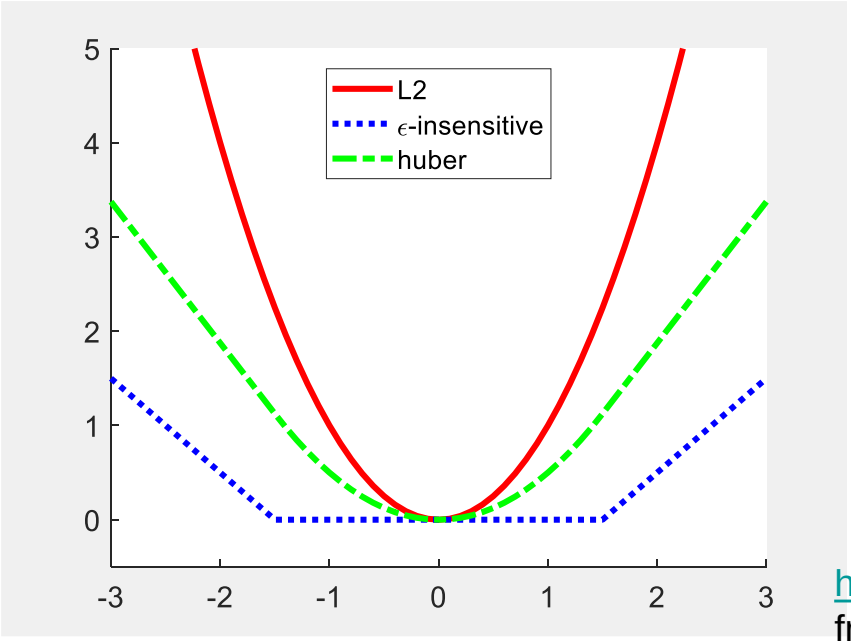
SVMs for Regression

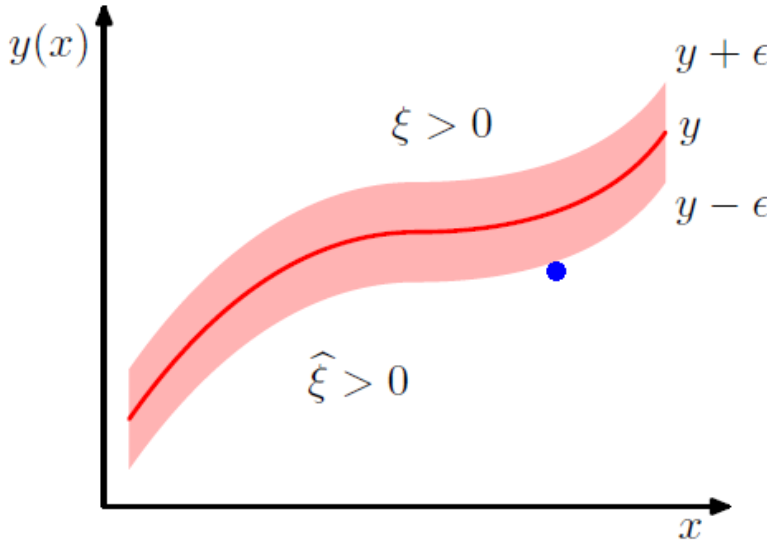
- In our approach to regression, $\boldsymbol{w}$ depends on all training inputs. We are here looking for sparse approximations.

- The ϵ –insensitive loss function: $L_\varepsilon(y,\hat{y}) \equiv \begin{cases} 0 & if \ \ |y-\hat{y}| < \varepsilon \\ |y-\hat{y}| - \epsilon & if \ \ |y-\hat{y}| > \varepsilon \end{cases}$

  leading to the following objective function:

$$\mathrm{J} = \mathrm{C}\sum_{i=1}^{N} L_\varepsilon(y_i,\hat{y}_i) + \frac{1}{2}\|\boldsymbol{w}\|^2,\ \hat{y}_i = f(\boldsymbol{x}_i) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + w_0, C = 1/\lambda$$





huberLossDemo.m
from PMTK3

SVMs for Regression

- The condition for a target point to lie inside the ε-tube is that $y_n - \varepsilon \leq t_n \leq y_n + \varepsilon$ where $y_n = y(\boldsymbol{x}_n)$. Introducing the slack variables allows points to lie outside the tube provided the slack variables are nonzero, and the corresponding conditions are

$$t_n \leq y(\mathbf{x}_n) + \epsilon + \xi_n$$
$$t_n \geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n$$

- The error function for support vector regression can then be written a

$$C \sum_{n=1}^{N} (\hat{\xi}_n + \xi_n) + \frac{1}{2} \|\boldsymbol{w}\|^2$$

which must be minimized subject to the constraints above as well as the constraints for the Lagrange multipliers $a_n \geq 0$ and $\hat{a}_n \geq 0$, $\mu_n \geq 0$ and $\hat{\mu}_n \geq 0$.

$$L = C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2} \|\boldsymbol{w}\|^2 - \sum_{n-1}^{N} (\mu_n \xi_n + \hat{\mu}_n \hat{\xi}_n)$$
$$- \sum_{n=1}^{N} a_n (\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^{N} \widehat{a_n} (\epsilon + \hat{\xi}_n - y_n + t_n)$$

SVMs for Regression

$$L = C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{n-1}^{N}(\mu_n\xi_n + \hat{\mu}_n\hat{\xi}_n) - \sum_{n=1}^{N} a_n(\epsilon + \xi_n + y_n - t_n) - \sum_{n=1}^{N}\widehat{a_n}(\epsilon + \hat{\xi}_n - y_n - t_n)$$

- We substitute $y(\boldsymbol{x})$ and set the derivatives wrt $\boldsymbol{w}, b, \xi_n, \hat{\xi}_n$ equal to 0.

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \rightarrow \boldsymbol{w} = \sum_{n=1}^{N}(a_n - \hat{a}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)$$

$$\frac{\partial L}{\partial b} = 0 \rightarrow \sum_{n=1}^{N}(a_n - \hat{a}_n) = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \rightarrow a_n + \mu_n = C$$

$$\frac{\partial L}{\partial \hat{\xi}_n} = 0 \rightarrow \hat{a}_n + \hat{\mu}_n = C$$

SVM Regression

- Eliminating the corresponding variables, we arrive at the dual problem

$$\tilde{L}(\boldsymbol{a}, \hat{\boldsymbol{a}}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$
$$-\varepsilon\sum_{n=1}^{N}(a_n + \hat{a}_n) + \sum_{n=1}^{N}(a_n - \hat{a}_n)t_n$$

- From $a_n + \mu_n = C$, $\hat{a}_n + \hat{\mu}_n = C$ and $\mu_n \geq 0$ and $\hat{\mu}_n \geq 0$, the following box constraints are applied

$$0 \leq a_n \leq C \text{ and } 0 \leq \hat{a}_n \leq C$$

in addition to $a_n \geq 0$ and $\hat{a}_n \geq 0$, $\mu_n \geq 0$ and $\hat{\mu}_n \geq 0$

- The predictions for new inputs are made using:

$$y(\boldsymbol{x}) = \sum_{n=1}^{N}(a_n - \hat{a}_n)k(\boldsymbol{x}, \boldsymbol{x}_n) + b$$

KKT Conditions

- The KKT conditions are:

$$a_n(\varepsilon + \xi_n + y_n - t_n) = 0$$

$$\hat{a}_n\left(\varepsilon + \hat{\xi}_n - y_n + t_n\right) = 0$$

$$(C - a_n)\xi_n = 0$$

$$(C - \hat{a}_n)\hat{\xi}_n = 0$$

- We note that a_n can only be nonzero if $\varepsilon + \xi_n + y_n - t_n = 0$, which implies that the data point either lies on the upper boundary of the ε-tube ($\xi_n = 0$) or lies above the upper boundary ($\xi_n = 0$).

- Similarly, a nonzero value for $\hat{a}_n$ implies $\varepsilon + \hat{\xi}_n - y_n + t_n = 0$ and such points must lie either on or below the lower boundary of the ϵ-tube.

KKT Conditions

- The KKT conditions are:

$$a_n(\varepsilon + \xi_n + y_n - t_n) = 0$$

$$\hat{a}_n\left(\varepsilon + \hat{\xi}_n - y_n + t_n\right) = 0$$

$$(C - a_n)\xi_n = 0$$

$$(C - \hat{a}_n)\hat{\xi}_n = 0$$

- Furthermore, the two constraints $\varepsilon + \xi_n + y_n - t_n = 0$ and $\varepsilon + \hat{\xi}_n - y_n + t_n = 0$ are incompatible, as is easily seen by adding them together and noting that ξ_n and $\hat{\xi}_n$ are nonnegative while ε is strictly positive, and so for every data point $\boldsymbol{x}_n$, either a_n or $\hat{a}_n$ (or both) must be zero.
- The support vectors are those data points that contribute to predictions i.e. those for which either $a_n \neq 0$ or $\hat{a}_n \neq 0$. These are points that lie on the boundary of the -tube or outside the tube.

KKT Conditions

- All points within the tube have $a_n = \hat{a}_n = 0$, i.e. we obtain again a sparse solution.

- The parameter *b* can be found by considering a data point for which $0 < a_n < C$, which from $(C - a_n)\xi_n = 0$ must have $\xi_n = 0$, and from $a_n(\varepsilon + \xi_n + y_n - t_n) = 0$ must therefore satisfy $\varepsilon + y_n - t_n = 0$. Using $y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) + b$ and $\boldsymbol{w} = \sum_{n=1}^{N}(a_n - \hat{a}_n)\boldsymbol{\phi}(\boldsymbol{x}_n)$ and solving for *b*, we obtain

$$b = t_n - \varepsilon - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x_n}) = t_n - \varepsilon - \sum_{m=1}^{N}(a_m - \hat{a}_m)k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

KKT Conditions

- There is an alternative formulation of the SVM for regression. Instead of fixing the width ε of the insensitive region, we fix instead a parameter ν that bounds the fraction of points lying outside the cube. We maximize:

$$\tilde{L}(\boldsymbol{a}, \hat{\boldsymbol{a}}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)k(\boldsymbol{x}_n, \boldsymbol{x}_m) + \sum_{n=1}^{N}(a_n - \hat{a}_n)t_n$$

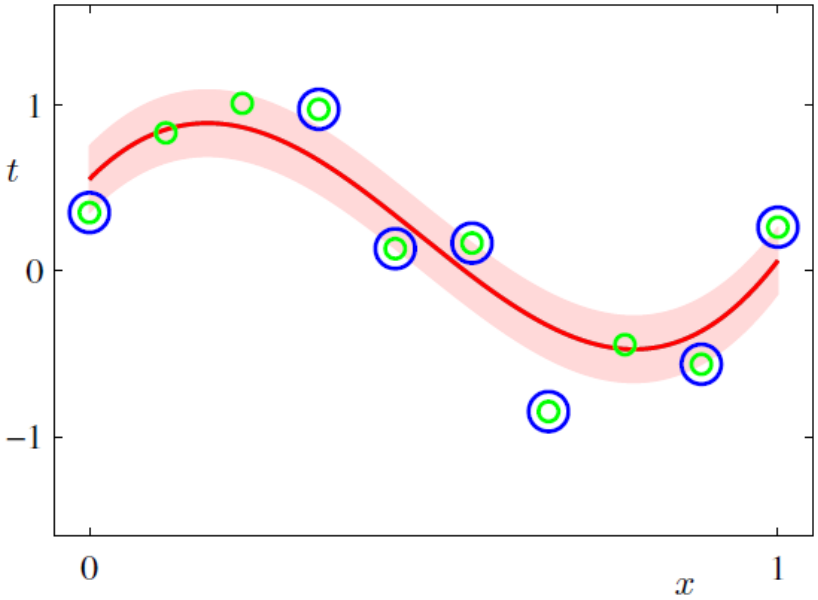
subject to:

$$0 \le a_n \le \frac{C}{N}, 0 \le \hat{a}_n \le \frac{C}{N}, \sum_{n=1}^{N}(a_n - \hat{a}_n) = 0, \sum_{n=1}^{N}(a_n + \hat{a}_n) \le \nu C$$

- It can be shown that there are at most νN data points falling outside the insensitive tube, while at least νN data points are support vectors and so lie either on the tube or outside it.

ν-SVM for Regression

- The use of a support vector machine to solve a regression problem is illustrated below. Here the parameters ν and C have been chosen by hand. Their values would typically be determined by cross validation.



- Illustration of the *v*-SVM for regression applied to the sinusoidal synthetic data set using Gaussian kernels. The predicted regression curve is shown by the red line, and the -insensitive tube corresponds to the shaded region. Also, the data points are shown in green, and those with support vectors are indicated by blue circles.

Computational Learning Theory

- SVMs have been motivated and analyzed using *computational learning theory*, also sometimes called *statistical learning theory.*
- Valiant formulated the *probably approximately correct*, or PAC, learning framework.
- The goal of PAC is to learn how large a data set needs to be in to give good generalization. It also gives bounds for the cost of learning.
- Suppose a data set $\mathcal{D}$ of size N is drawn from $p(\boldsymbol{x}, \boldsymbol{t})$ where $\boldsymbol{x}$ is the input variable and $\boldsymbol{t}$ represents the class label, and restrict to 'noise free' situations in which $\boldsymbol{t}$ are determined by some (unknown) deterministic function $\boldsymbol{t} = \boldsymbol{g}(\boldsymbol{x})$.

- Valiant, L. G. (1984). A theory of the learnable. *Communications of the Association for Computing Machinery* **27**, 1134–1142.
- Anthony, M. and N. Biggs (1992). *An Introduction to Computational Learning Theory*. Cambridge University Press.
- Kearns, M. J. and U. V. Vazirani (1994). *An Introduction to Computational Learning Theory*. MIT Press.
- Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer.
- Vapnik, V. N. (1998). *Statistical learning theory*.Wiley.
- McAllester, D. A. (2003). PAC-Bayesian stochastic model selection. *Machine Learning* **51**(1), 5–21.

Computational Learning Theory

- In PAC learning we say that a function $\boldsymbol{f}\ (\boldsymbol{x}; \mathcal{D})$, drawn from a space $\mathcal{F}$ of such functions on the basis of the training set $\mathcal{D}$, has good generalization if its expected error rate is below some pre-specified threshold , so that

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{t}}[\mathbb{I}\ (\boldsymbol{f}\ (\boldsymbol{x}; \mathcal{D}) = \ \boldsymbol{t})] < \ \varepsilon$$

- The quantity on the left-hand side is a random variable, because it depends on the training set $\mathcal{D}$, and the PAC framework requires that the Eq. above holds with probability greater than $1 - \delta$, for a data set $\mathcal{D}$ drawn randomly from $p(\boldsymbol{x}, \boldsymbol{t})$.

- Here δ is another pre-specified parameter, and the terminology 'probably approximately correct' comes from the requirement that with high probability (greater than $1 - \delta$), the error rate be small (less than ε).

Computational Learning Theory

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{t}}[\mathbb{I}\,(\boldsymbol{f}\,(\boldsymbol{x};\mathcal{D}) = \boldsymbol{t})] < \varepsilon$$

- For a given choice of model space $\mathcal{F}$, and for given parameters and δ, PAC learning aims to provide bounds on the minimum size N of data set needed to meet this criterion.

- A key quantity in PAC learning is the *Vapnik-Chervonenkis dimension*, or VC dimension, which provides a measure of the complexity of a space of functions, and which allows the PAC framework to be extended to spaces containing an infinite number of functions.

- The bounds derived within the PAC framework are often described as worst case, because they apply to *any* choice for the distribution $p(\boldsymbol{x}, \boldsymbol{t})$, so long as both the training and the test examples are drawn (independently) from the same distribution, and for *any* choice for the function $\boldsymbol{f}\,(\boldsymbol{x})$ so long as it belongs to $\mathcal{F}$.

Computational Learning Theory

- In real-world applications of machine learning, we deal with distributions that have significant regularity, for example in which large regions of input space carry the same class label.

- The PAC bounds are very conservative, in other words they strongly over-estimate the size of data sets required to achieve a given generalization performance. For this reason, PAC bounds have found few, if any, practical applications.

- One attempt to improve the tightness of the PAC bounds is the *PAC-Bayesian* framework, which considers a distribution over the space $\mathcal{F}$ of functions, somewhat analogous to the prior in a Bayesian treatment. This still considers any possible choice for $p(\boldsymbol{x}, \boldsymbol{t})$, and so although the bounds are tighter, they are still very conservative.

- McAllester, D. A. (2003). PAC-Bayesian stochastic model selection. *Machine Learning* **51**(1), 5–21.

Relevance Vector Machines

Relevance Vector Machines

- Relevance Vector Machines (RVM) were introduced by M. Tipping in Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. Journal of Machine Learning Research 1, 211–244.
- It is a *Bayesian regression* technique on a *generalized linear model* that produces *sparse representations* (in the sense that many of the coefficients of the generalized linear model turn out to be zero).
- We will first describe it on one output dimension but can generalize it to many dimensions.
- One can couple it with an adaptive input decomposition algorithm and apply it to uncertainty quantification tasks. See I. Bilionis and N. Zabaras. *Multidimensional adaptive relevance vector machines for uncertainty quantification*. SIAM Journal for Scientific Computing, Vol. 34, No. 6, pp. B881–B908 2012 for further details.

Relevance Vector Machines

- RVM addresses many of the limitations of SVM:
  - SVM outputs are decisions rather than posterior probabilities.
  - Extention to K>2 classes in SVM is problematic.
  - There are complexity parameters in SVM to be found by cross-validation.
  - Predictions in SVM are based on kernel functions centered at the training data that are required to be positive definite.

RVM: Likelihood

- We assume that we observe the data:

$$\mathcal{D} = \left\{\boldsymbol{x}^{(i)}, y^{(i)} = f\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{n},$$

and that we wish to learn the function $f(\cdot)$.

- Pick a set of M basis functions $\boldsymbol{\phi}: \boldsymbol{R}^k \rightarrow \boldsymbol{R}^M$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \left(\phi_1(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x})\right).$$

- We will approximate $f(\cdot)$ by:

$$\hat{f}(x; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + \epsilon = \sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x}) + \epsilon,$$

where ϵ is a random variable representing noise.

- For simplicity, we choose to work with Gaussian noise with zero mean and inverse variance β, so the **likelihood** is:

$$p(t|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t|\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}), \beta^{-1})$$

RVM: Prior

- The essence of RVM is the following **prior on the weights**:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{j=1}^{M} p\left(w_j|\alpha_j\right),$$

with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)$ and:

$$p\left(w_j|\alpha_j\right) = \mathcal{N}\left(w_j|0, \alpha_j^{-1}\right).$$

- The characteristic of this prior is that:

$$\lim_{\alpha_j \to \infty} p\left(w_j|\alpha_j\right) = \delta\left(w_j\right).$$

- This means that the j-th basis function can be removed from the model if its α_j is very big. This is because, its weights will be very sharply picked about zero.

- To complete the model, we also need to assign a prior $p(\beta, \boldsymbol{\alpha})$. We manage to get rid of it by employing the *evidence approximation.*

RVM: Posterior

- The **posterior of the weights** is:

$$p(\boldsymbol{w}|\mathcal{D},\beta,\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m},\boldsymbol{\Sigma}),$$

where:

$$\boldsymbol{\Sigma} = \left(\mathrm{diag}(\boldsymbol{\alpha}) + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1},$$

$$\boldsymbol{m} = \beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{t},$$

with $\boldsymbol{t} = \left(t^{(1)},\ldots,t^{(n)}\right)$ is the *vector of observations*, $\boldsymbol{\Phi} \in \boldsymbol{R}^{N\times M}$ is the *design matrix*:

$$\Phi_{ij} = \phi_j\left(\boldsymbol{x}^{(i)}\right),$$

and $\mathrm{diag}(\boldsymbol{\alpha})$ is a diagonal matrix with $\boldsymbol{\alpha}$ on the diagonal.

Proof:

First, notice that we can write:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{j=1}^{M} p\left(w_j|\alpha_j\right) = \prod_{j=1}^{M} \mathcal{N}\left(w_j|0,\alpha_j^{-1}\right) = \mathcal{N}_M(\boldsymbol{w}|\mathrm{diag}(\boldsymbol{\alpha})^{-1}).$$

RVM: Posterior

Proof (continuation):

Also, notice that the likelihood of the data (under an independence assumption) is given by:

$$p(\mathcal{D}|\boldsymbol{w},\beta)=\prod_{i=1}^{n}\mathcal{N}\left(t^{(i)}\middle|\boldsymbol{w}^T\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right),\beta^{-1}\right)=\mathcal{N}_N(\boldsymbol{y}|\boldsymbol{\Phi}\boldsymbol{w},\beta^{-1}\boldsymbol{I}_N),$$

where $\boldsymbol{I}_N$ is the N-dimensional unit matrix and we have used the fact that:

$$\sum_{i=1}^{N}\|\,\boldsymbol{w}^T\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)-t^{(i)}\,\|^2=\sum_{i=1}^{N}\left(\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\boldsymbol{w}-t^{(i)}\right)^T\left(\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\boldsymbol{w}-t^{(i)}\right)$$
$$=(\boldsymbol{\Phi}\boldsymbol{w}-\mathbf{y})^{\mathrm{T}}(\boldsymbol{\Phi}\boldsymbol{w}\ -\mathbf{y}).$$

Finally, we have:

$$p(\boldsymbol{w}|\mathcal{D},\boldsymbol{\alpha},\beta)\propto p(\mathcal{D}|\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha})=\mathcal{N}_N(\boldsymbol{y}|\boldsymbol{\Phi}\boldsymbol{w},\beta^{-1}\boldsymbol{I}_N)\mathcal{N}_M(\boldsymbol{w}|\mathrm{diag}(\boldsymbol{\alpha})^{-1}).$$

RVM: Posterior

Proof (continuation):

Remember that the posterior is:

$$p(\boldsymbol{w}|\mathcal{D},\boldsymbol{\alpha},\beta) \propto p(\mathcal{D}|\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathcal{N}_N(\boldsymbol{y}|\boldsymbol{\Phi}\mathbf{w},\beta^{-1}\boldsymbol{I}_N)\mathcal{N}_M(\boldsymbol{w}|\mathrm{diag}(\boldsymbol{\alpha})^{-1}).$$

Now, we just have to look at it and complete the square…

You can simply drop this guy!

We have for the exponential inside the right hand side:

$$(\boldsymbol{\Phi w}-\boldsymbol{t})^{\mathrm{T}}(\beta\boldsymbol{I}_N)(\boldsymbol{\Phi w}-\boldsymbol{t})+\boldsymbol{w}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{w} =$$
$$\boldsymbol{w}^T\left(\beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)\boldsymbol{w}\ -2\boldsymbol{w}^{\mathrm{T}}\left(\beta\boldsymbol{\Phi}^{\mathrm{T}}\right)\boldsymbol{y}+\beta\boldsymbol{t}^T\boldsymbol{t}+\boldsymbol{w}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{w} =$$
$$\boldsymbol{w}^T\left(\mathrm{diag}(\boldsymbol{\alpha})+\beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)\boldsymbol{w}\ -2\boldsymbol{w}^{\mathrm{T}}\left(\beta\boldsymbol{\Phi}^{\mathrm{T}}\right)\mathbf{t} =$$
$$\boldsymbol{w}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{w}\ -2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left(\beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\right)\mathbf{t}.$$

From which the result follows directly. End of Proof.

Important remark: When completing the square for a distribution, you can subtract, add or even ignore anything that does not depend on the random variable whose posterior you are calculating!

Or using Results for Linear Gaussian Models

$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}, \Lambda^{-1}\right)$$

$$p(\mathbf{y} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right)$$

- For the above linear Gaussian model, the following hold:

$$p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\Lambda^{-1}\mathbf{A}^T\right)$$

$$p(\mathbf{x} \mid \mathbf{y}) = \mathcal{N}\left(\mathbf{x} \mid \left(\Lambda + \mathbf{A}^T\mathbf{L}\mathbf{A}\right)^{-1}\left(\Lambda\boldsymbol{\mu} + \mathbf{A}^T\mathbf{L}(\mathbf{y} - \mathbf{b})\right), \left(\Lambda + \mathbf{A}^T\mathbf{L}\mathbf{A}\right)^{-1}\right)$$

- For our problem $\boldsymbol{x} \to \boldsymbol{w}, \boldsymbol{\mu} \to \mathbf{0}, \boldsymbol{\Lambda} \to \text{diag}(\boldsymbol{\alpha}), \boldsymbol{y} \to \boldsymbol{t}, \boldsymbol{A} \to \boldsymbol{\Phi}, \boldsymbol{b} \to \mathbf{0}, \boldsymbol{L} = \boldsymbol{\beta}\boldsymbol{I}_N$

$$p(\boldsymbol{t}|\boldsymbol{w}) = \mathcal{N}_N(\boldsymbol{y}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\boldsymbol{I}_N), p(\boldsymbol{w}) = \mathcal{N}_M\left(\boldsymbol{w}|\text{diag}(\boldsymbol{\alpha})^{-1}\right)$$

and: $p(\boldsymbol{w}|\mathcal{D}, \beta, \boldsymbol{\alpha}) = \mathcal{N}\left(\boldsymbol{w}\middle|\beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{t}, \underbrace{\left(\text{diag}(\boldsymbol{\alpha}) + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}}_{\boldsymbol{\Sigma}}\right)$

RVM: Predictive Distribution

The **predictive** distribution is easily found as follows:

$$p(t|\boldsymbol{x},\boldsymbol{X},\boldsymbol{t},\boldsymbol{\alpha},\beta) = \int p(t|\boldsymbol{x},\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{t},\boldsymbol{\alpha},\boldsymbol{\beta})d\boldsymbol{w}$$
$$= \mathcal{N}(t|\boldsymbol{m}^T\boldsymbol{\phi}(\boldsymbol{x}),\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\Sigma}\boldsymbol{\phi}(\boldsymbol{x}) + \beta^{-1}).$$

So, we have the *predictive mean (**m** being the posterior mean)*:

$$\boldsymbol{\mu}(\boldsymbol{x}) = \boldsymbol{m}^T\boldsymbol{\phi}(\boldsymbol{x}),$$

and the predictive variance:

$$\sigma^2(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\Sigma}\boldsymbol{\phi}(\boldsymbol{x}) + \beta^{-1}.$$

Proof:

Complete the square….

End of proof.

RVM: Evidence Approximation

- We will select the hyper-parameters $\boldsymbol{\alpha}$ and β by maximizing employing the *evidence approximation* (otherwise known as *maximizing the marginal likelihood*). We start by motivating it.

- Assume that we have a prior for $\boldsymbol{\alpha}$ and β, $p(\boldsymbol{\alpha}, \beta)$. Then the posterior of these parameters is:

$$p(\boldsymbol{\alpha}, \beta|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\alpha}, \beta)p(\boldsymbol{\alpha}, \beta) = \int p(\mathcal{D}|\boldsymbol{w}, \beta)p(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w}\, p(\boldsymbol{\alpha}, \beta).$$

Marginal Likelihood (or Evidence)

- Under the evidence approximation, we assume that:
  - $p(\boldsymbol{\alpha}, \beta)$ is relatively flat.
  - $p(\mathcal{D}|\boldsymbol{\alpha}, \beta)$ has very sharp maxima.

Then, we may obtain point estimates of the hyper-parameters by identifying these maxima.

RVM: Evidence

- To carry out the evidence approximation, we must obtain an analytic expression for the marginal likelihood. This is possible for RVM:

$$\mathcal{E}(\boldsymbol{\alpha}, \beta) = \log\big(p(\mathcal{D}|\boldsymbol{\alpha}, \beta)\big) = -\frac{1}{2}[N\log(2\pi) + \log|\boldsymbol{C}| + \boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t}],$$

where $\boldsymbol{C} \in \boldsymbol{R}^{N\times N}$, is given by:

$$\boldsymbol{C} = \beta^{-1}\boldsymbol{I}_N + \boldsymbol{\Phi}\mathrm{diag}(\boldsymbol{\alpha})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}.$$

Proof:

This does require some more work that we avoided doing before. We start again from:

$$p(\mathcal{D}|\boldsymbol{w}, \beta)p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathcal{N}_N(\boldsymbol{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\boldsymbol{I}_N)\mathcal{N}_M(\boldsymbol{w}|\mathrm{diag}(\boldsymbol{\alpha})^{-1}),$$

and we will complete the square making sure this time that we keep track of all terms that depend on $\boldsymbol{t}$.

RVM: Evidence

Proof (continuation):

Remember, that we want to integrate $\boldsymbol{w}$ out of:

$$p(\mathcal{D}|\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathcal{N}_n(\boldsymbol{t}|\boldsymbol{\Phi}\mathbf{w},\beta^{-1}\boldsymbol{I}_N)\mathcal{N}_M(\boldsymbol{w}|\mathrm{diag}(\boldsymbol{\alpha})^{-1}).$$

Again, we look at whatever is insider the exponential of the right-hand side and attempt to complete the square. This time, we already know the answer, but we need to keep track of $\boldsymbol{t}$ explicitly:

$$(\boldsymbol{\Phi w}-\boldsymbol{t})^{\mathrm{T}}(\beta\boldsymbol{I}_N)(\boldsymbol{\Phi w}-\boldsymbol{t})+\boldsymbol{w}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{w}=$$
$$\boldsymbol{w}^T\left(\beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)\boldsymbol{w}-2\boldsymbol{w}^{\mathrm{T}}\left(\beta\boldsymbol{\Phi}^{\mathrm{T}}\right)\boldsymbol{t}+\beta\boldsymbol{t}^T\boldsymbol{t}+\boldsymbol{w}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{w}=$$
$$\boldsymbol{w}^T\left(\mathrm{diag}(\boldsymbol{\alpha})+\beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)\boldsymbol{w}-2\boldsymbol{w}^{\mathrm{T}}\left(\beta\boldsymbol{\Phi}^{\mathrm{T}}\right)\boldsymbol{t}+\beta\boldsymbol{t}^T\mathbf{t}=$$
$$\boldsymbol{w}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{w}-2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left(\beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\right)\boldsymbol{t}+\beta\boldsymbol{t}^T\boldsymbol{t}=$$
$$\boldsymbol{w}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{w}-2\boldsymbol{w}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{m}+\boldsymbol{m}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{m}-\boldsymbol{m}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{m}+\beta\boldsymbol{t}^T\boldsymbol{t}=$$
$$(\boldsymbol{w}-\boldsymbol{m})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{w}-\boldsymbol{m})+(\beta\boldsymbol{t}^T\boldsymbol{t}-\boldsymbol{m}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{m}).$$

RVM: Evidence

Proof (continuation):

We can now write:

$$p(\mathcal{D}|\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathcal{N}_N(\boldsymbol{y}|\boldsymbol{\Phi}\mathbf{w},\beta^{-1}\boldsymbol{I}_N)\mathcal{N}_M(\boldsymbol{w}|\text{diag}(\boldsymbol{\alpha})^{-1}) =$$

$$(2\pi)^{-\frac{N}{2}}\beta^{\frac{N}{2}}(2\pi)^{-\frac{M}{2}}|\text{diag}(\boldsymbol{\alpha})|^{\frac{1}{2}}\exp\left\{-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{m})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{w}-\boldsymbol{m})\right.$$

RVM: Two useful matrix identities

- *Woodbury Matrix Identity*:

$$(\boldsymbol{A}+\boldsymbol{UCV})^{-1}=\boldsymbol{A}^{-1}-\boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1}+\boldsymbol{VA}^{-1}\boldsymbol{U})^{-1}\boldsymbol{VA}^{-1}.$$

- *Matrix Determinant Lemma*:

$$|\boldsymbol{A}+\boldsymbol{UWV}^{T}|=|\boldsymbol{W}^{-1}+\boldsymbol{V}^{T}\boldsymbol{A}^{-1}\boldsymbol{U}||\boldsymbol{W}||\boldsymbol{A}|.$$

You do not have to remember these, but you must be aware of there existence. Sometimes, they are the only route to analytic progress!

Or using Results for Linear Gaussian Models

$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \mid \boldsymbol{\mu}, \Lambda^{-1}\right)$$

$$p(\mathbf{y} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right)$$

- For the above linear Gaussian model, the following hold:

$$p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\Lambda^{-1}\mathbf{A}^T\right)$$

$$p(\mathbf{x} \mid \mathbf{y}) = \mathcal{N}\left(\mathbf{x} \mid \left(\Lambda + \mathbf{A}^T\mathbf{L}\mathbf{A}\right)^{-1}\left(\Lambda\boldsymbol{\mu} + \mathbf{A}^T\mathbf{L}(\mathbf{y} - \mathbf{b})\right), \left(\Lambda + \mathbf{A}^T\mathbf{L}\mathbf{A}\right)^{-1}\right)$$

- For our problem $\boldsymbol{x} \to \boldsymbol{w}, \boldsymbol{\mu} \to \mathbf{0}, \boldsymbol{\Lambda} \to \text{diag}(\boldsymbol{\alpha}), \boldsymbol{y} \to \boldsymbol{t}, \boldsymbol{A} \to \boldsymbol{\Phi}, \boldsymbol{b} \to \mathbf{0}, \boldsymbol{L} = \boldsymbol{\beta}\boldsymbol{I}_N$

$$p(\boldsymbol{t}|\boldsymbol{w}) = \mathcal{N}_N(\boldsymbol{y}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\boldsymbol{I}_N), p(\boldsymbol{w}) = \mathcal{N}_M\left(\boldsymbol{w}|\text{diag}(\boldsymbol{\alpha})^{-1}\right)$$

and: $p(\boldsymbol{t}|\beta, \boldsymbol{\alpha}) = \mathcal{N}\left(\boldsymbol{w}\middle|\mathbf{0}, \beta^{-1}\boldsymbol{I}_N + \boldsymbol{\Phi}\text{diag}(\boldsymbol{\alpha})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\right)$

RVM: Evidence Approximation

- From our earlier derivation:

$$p(\mathcal{D}|\boldsymbol{\alpha},\beta)=(2\pi)^{-\frac{N}{2}}\beta^{\frac{N}{2}}|\text{diag}(\boldsymbol{\alpha})|^{\frac{1}{2}}\exp\left\{-\frac{1}{2}\boldsymbol{t}^T(\beta\boldsymbol{I}_N-\beta^2\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T)\boldsymbol{t}\right\}|\boldsymbol{\Sigma}|^{\frac{1}{2}}$$

where $\boldsymbol{\Sigma}=\left(\text{diag}(\boldsymbol{\alpha})+\beta\boldsymbol{\Phi}^{\text{T}}\boldsymbol{\Phi}\right)^{-1},\ \boldsymbol{m}=\beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\text{T}}\boldsymbol{t}$

- Noting that $\boldsymbol{t}^T\beta^2\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\boldsymbol{t}=\boldsymbol{m}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{m}$, can simplify as:

$$p(\mathcal{D}|\boldsymbol{\alpha},\beta)=(2\pi)^{-\frac{N}{2}}\beta^{\frac{N}{2}}|\text{diag}(\boldsymbol{\alpha})|^{\frac{1}{2}}\exp\left\{-\frac{1}{2}\left(\beta\boldsymbol{t}^T\boldsymbol{t}-\boldsymbol{m}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{m}\right)\right\}|\boldsymbol{\Sigma}|^{\frac{1}{2}}$$

- Taking derivatives wrt α_i of $\ln p(\mathcal{D}|\boldsymbol{\alpha},\beta)$ and using $\frac{\partial \ln|\boldsymbol{\Sigma}|}{\partial\alpha_i}=Tr\left(\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\alpha_i}\right)=-Tr\left(\boldsymbol{\Sigma}\frac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\alpha_i}\right)=-\Sigma_{ii}$, $\boldsymbol{m}^T\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{m}}{\partial\alpha_i}=-\boldsymbol{m}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}\beta\frac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\alpha_i}\boldsymbol{\Phi}^{\text{T}}\boldsymbol{t}=-m_i^2$ and setting equal to 0:

$$\frac{1}{2\alpha_i}+\frac{1}{2}m_i^2-m_i^2-\frac{1}{2}\Sigma_{ii}=0\rightarrow\alpha_i=\frac{1-\alpha_i\Sigma_{ii}}{m_i^2}\equiv\frac{\gamma_i}{m_i^2}$$

RVM: Evidence Approximation

$$p(\mathcal{D}|\boldsymbol{\alpha},\beta) = (2\pi)^{-\frac{N}{2}}\beta^{\frac{N}{2}}|\text{diag}(\boldsymbol{\alpha})|^{\frac{1}{2}}\exp\left\{-\frac{1}{2}\left(\beta\boldsymbol{t}^T\boldsymbol{t}-\boldsymbol{m}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{m}\right)\right\}|\boldsymbol{\Sigma}|^{\frac{1}{2}}$$

- Taking derivatives wrt β of $\ln p(\mathcal{D}|\boldsymbol{\alpha},\beta)$ and using $\frac{\partial \ln|\Sigma|}{\partial\beta} = Tr\left(\boldsymbol{\Sigma}^{-1}\frac{\partial\boldsymbol{\Sigma}}{\partial\beta}\right) = -Tr\left(\boldsymbol{\Sigma}\frac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\beta}\right) = -Tr\left(\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)$, and $\boldsymbol{m} = \beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{t}$, setting equal to 0:

$$\frac{N}{2\beta}-\frac{1}{2}\boldsymbol{t}^T\boldsymbol{t}+\frac{1}{2}\boldsymbol{m}^T\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\boldsymbol{m}+\boldsymbol{m}^T\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{t}-\frac{1}{2}tr\left(\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)=0 \rightarrow \left(\beta^{(new)}\right)^{-1} = \frac{\|\boldsymbol{t}-\boldsymbol{\Phi}\boldsymbol{m}\|^2}{N-\sum\gamma_i}$$

- Here, we used

$$\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}+\beta^{-1}\boldsymbol{\Sigma}\text{diag}(\boldsymbol{\alpha})-\beta^{-1}\boldsymbol{\Sigma}\text{diag}(\boldsymbol{\alpha}) =$$
$$= \boldsymbol{\Sigma}\left(\text{diag}(\boldsymbol{\alpha})+\beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)\beta^{-1}-\beta^{-1}\boldsymbol{\Sigma}\text{diag}(\boldsymbol{\alpha})$$
$$= \left(\text{diag}(\boldsymbol{\alpha})+\beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\left(\text{diag}(\boldsymbol{\alpha})+\beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)\beta^{-1}-\beta^{-1}\boldsymbol{\Sigma}\text{diag}(\boldsymbol{\alpha})$$
$$= \beta^{-1}(\mathbf{I}-\text{diag}(\boldsymbol{\alpha})\boldsymbol{\Sigma})$$

and thus $tr\left(\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right) = \beta^{-1}\sum\gamma_i$

RVM: Evidence Approximation

$$\alpha_i = \frac{1 - \alpha_i \Sigma_{ii}}{m_i^2} \equiv \frac{\gamma_i}{m_i^2}, \quad \left(\beta^{(new)}\right)^{-1} = \frac{\|\boldsymbol{t} - \boldsymbol{\Phi m}\|^2}{N - \sum \gamma_i}$$

- Learning therefore proceeds by
  - ✓ choosing initial values for $\boldsymbol{\alpha}$ and β,
  - ✓ evaluating the mean and covariance of the posterior using $\boldsymbol{\Sigma} = \left(\text{diag}(\boldsymbol{\alpha}) + \beta \boldsymbol{\Phi}^{\text{T}} \boldsymbol{\Phi}\right)^{-1}$, $\boldsymbol{m} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^{\text{T}} \boldsymbol{t}$, respectively, and then
  - ✓ alternately re-estimating the hyperparameters, using the Eqs. above and
  - ✓ iterating until a suitable convergence criterion is satisfied.

- The second alternative approach is to use the EM algorithm. These two approaches to finding the values of the hyperparameters that maximize the evidence are formally equivalent. Numerically, however, it is found that the direct optimization approach gives somewhat faster convergence (Tipping, 2001).

RVM: Fast training algorithm

- We will develop a **fast algorithm** for **maximizing the evidence**.
- We follow closely the developments in *Tipping, M. E. and A. C. Faul (2003). Fast marginal likelihood maximisation for sparse Bayesian models. In C. M. Bishop and B. J. Frey (Eds.), Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, Key West, FL, Jan 3-6*.
- There are some typos in the paper.
- We will achieve this by asking the following question:

Keeping β fixed and given some values for α, what is the action on a single basis function that gives the maximum increase in evidence?

This will require a very thorough examination of the evidence function.

- A Matlab implementation for the one dimensional case can be found here.

RVM: Splitting the evidence

- Some notation:
  - Subscript "$-i$": Wherever we see this applied to a vector or matrix, it gives the corresponding vector or matrix with the influence of the i –th basis function removed.
  - $\boldsymbol{\alpha}_{-i} = (\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_m)$: All the hyper-parameters except the ones that pertain to basis function i.
  - $\boldsymbol{\Phi}_{-i}$: The design matrix with the column corresponding to basis function i removed.
  - $\boldsymbol{C}_{-i} = \beta^{-1}\boldsymbol{I}_N + \boldsymbol{\Phi}_{-\mathrm{i}}\mathrm{diag}(\boldsymbol{\alpha}_{-i})^{-1}\boldsymbol{\Phi}_{-i}^{\mathrm{T}}$: The matrix $\boldsymbol{C}$, without the influence of basis function i.
  - $\mathcal{E}(\boldsymbol{\alpha}_{-i}, \beta) = -\frac{1}{2}\left[n\log(2\pi) + \log|\boldsymbol{C}_{-i}| + \boldsymbol{y}^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{y}\right]$: The evidence when the i –th basis function has been removed.

RVM: Splitting the evidence

Given any basis function $i \in \{1, \dots, M\}$, the evidence can be split as:

$$\mathcal{E}(\boldsymbol{\alpha}, \beta) = \mathcal{E}(\boldsymbol{\alpha}_{-i}, \beta) + \epsilon(\alpha_i),$$

where

$$\epsilon(\alpha_i) = \frac{1}{2}\left(\log \alpha_i - \log(\alpha_i + s_i) + \frac{h_i^2}{\alpha_i + s_i}\right),$$

with

$$s_i = \boldsymbol{\varphi}_i^T \boldsymbol{C}_{-i}^{-1} \boldsymbol{\varphi}_i \quad \text{and} \quad h_i = \boldsymbol{\varphi}_i^T \boldsymbol{C}_{-i}^{-1} \boldsymbol{t}.$$

Remark: Everything that relates to basis function i is here!

Proof:

We start by splitting $\boldsymbol{C}$:

$$\boldsymbol{C} = \beta^{-1}\boldsymbol{I}_N + \boldsymbol{\Phi}\text{diag}(\boldsymbol{\alpha})^{-1}\boldsymbol{\Phi}^{\mathrm{T}}$$

$$= \beta^{-1}\boldsymbol{I}_N + (\boldsymbol{\Phi}_{-\mathrm{i}} \quad \boldsymbol{\varphi}_i)\begin{pmatrix} \text{diag}(\boldsymbol{\alpha}_{-i})^{-1} & \boldsymbol{0}_{(\mathrm{M}-1)\times 1} \\ \boldsymbol{0}_{1\times(M-1)} & \alpha_i^{-1} \end{pmatrix}\begin{pmatrix} \boldsymbol{\Phi}_{-i}^T \\ \boldsymbol{\varphi}_i^T \end{pmatrix}$$

$$= \beta^{-1}\boldsymbol{I}_N + \boldsymbol{\Phi}_{-\mathrm{i}}\text{diag}(\boldsymbol{\alpha}_{-i})^{-1}\boldsymbol{\Phi}_{-\mathrm{i}}^{\mathrm{T}} + \alpha_i^{-1}\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^T$$

$$= \boldsymbol{C}_{-i} + \alpha_i^{-1}\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^T.$$

RVM: Splitting the evidence

Proof (continuation):

Remember that the evidence is:

$$\mathcal{E}(\boldsymbol{\alpha}, \beta) = -\frac{1}{2}[N\log(2\pi) + \log|\boldsymbol{C}| + \boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t}].$$

We will only make use of this:

$$\boldsymbol{C} = \boldsymbol{C}_{-i} + \alpha_i^{-1}\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^T,$$

and some identities.

$$(\boldsymbol{A} + \boldsymbol{UCV})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1} + \boldsymbol{VA}^{-1}\boldsymbol{U})^{-1}\boldsymbol{VA}^{-1}.$$

From the Woodbury identity, we get:

$$\boldsymbol{C}^{-1} = \boldsymbol{C}_{-i}^{-1} - \boldsymbol{C}_{-i}^{-1}\boldsymbol{\varphi}_i\left(\alpha_i + \boldsymbol{\varphi}_i^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{\varphi}_i\right)^{-1}\boldsymbol{\varphi}_i^T\boldsymbol{C}_{-i}^{-1}$$

$$= \boldsymbol{C}_{-i}^{-1} - \frac{\boldsymbol{C}_{-i}^{-1}\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^T\boldsymbol{C}_{-i}^{-1}}{\alpha_i + s_i}.$$

And we get:

$$\boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t} = \boldsymbol{t}^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{t} - \frac{\left(\boldsymbol{t}^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{\varphi}_i\right)\left(\boldsymbol{\varphi}_i^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{t}\right)}{\alpha_i + s_i} = \boldsymbol{t}^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{t} + \frac{h_i^2}{\alpha_i + s_i}.$$

RVM: Splitting the evidence

Proof (continuation):

Finally, we look at $|\boldsymbol{C}|$. Remember:

$$\boldsymbol{C} = \boldsymbol{C}_{-i} + \alpha_i^{-1}\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^T,$$

Using the matrix determinant lemma:

$$|\boldsymbol{A} + \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T| = |\boldsymbol{W}^{-1} + \boldsymbol{V}^T\boldsymbol{A}^{-1}\boldsymbol{U}||\boldsymbol{W}||\boldsymbol{A}|$$

$$\begin{aligned}|\boldsymbol{C}| &= \left|\alpha_i + \boldsymbol{\varphi}_i^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{\varphi}_i\right|\left|\alpha_i^{-1}\right||\boldsymbol{C}_{-i}| \\ &= \frac{|\boldsymbol{C}_{-i}|(\alpha_i + s_i)}{\alpha_i}.\end{aligned}$$

Or

$$\log|\boldsymbol{C}| = \log|\boldsymbol{C}_{-i}| - \log\alpha_i + \log(\alpha_i + s_i).$$

RVM: Splitting the evidence

Proof (continuation):

Now we combine everything. Remember that:

$$\boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t} = \boldsymbol{t}^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{t} + \frac{h_i^2}{\alpha_i + s_i},$$

and

$$\log|\boldsymbol{C}| = \log|\boldsymbol{C}_{-i}| - \log\alpha_i + \log(\alpha_i + s_i)\,.$$

Therefore, we have for the evidence:

$$\mathcal{E}(\boldsymbol{\alpha},\beta) = -\frac{1}{2}[N\log(2\pi) + \log|\boldsymbol{C}| + \boldsymbol{t}^T\boldsymbol{C}^{-1}\boldsymbol{t}]$$

$$= -\frac{1}{2}\left(N\log(2\pi) + \log|\boldsymbol{C}_{-i}| + \boldsymbol{t}^T\boldsymbol{C}_{-i}^{-1}\boldsymbol{t} + \frac{h_i^2}{\alpha_i + s_i} - \log\alpha_i + \log(\alpha_i + s_i)\right)$$

$$= \mathcal{E}(\boldsymbol{\alpha}_{-i},\beta) + \epsilon(\alpha_i).$$

End of proof.

RVM: Analysis of the evidence

We have shown that for any basis function $i \in \{1, \ldots, M\}$, the evidence can be split as:

$$\mathcal{E}(\boldsymbol{\alpha}, \beta) = \mathcal{E}(\boldsymbol{\alpha}_{-i}, \beta) + \epsilon(\alpha_i),$$

where

$$\epsilon(\alpha_i) = \frac{1}{2}\left(\log \alpha_i - \log(\alpha_i + s_i) + \frac{h_i^2}{\alpha_i + s_i}\right).$$

We will now analyze the $\epsilon(\alpha_i)$ term in more detail.

We compute its first derivative:

$$\frac{\partial \epsilon(\alpha_i)}{\partial \alpha_i} = \frac{s_i^2 - \alpha_i \theta_i}{2\alpha_i(\alpha_i + s_i)^2},$$

where

$$\theta_i = h_i^2 - s_i.$$

RVM: Analysis of the evidence

Setting

$$\frac{\partial \epsilon(\alpha_i)}{\partial \alpha_i} = \frac{s_i^2 - \alpha_i \theta_i}{2\alpha_i(\alpha_i + s_i)^2}$$

equal to zero, we have two possibilities:

- If $\theta_i > 0$, then

$$\alpha_i^* = \frac{s_i^2}{\theta_i},$$

is the unique maximum of $\epsilon(\alpha_i)$ (calculate the 2nd derivative to be convinced that it is indeed a maximum).

- If $\theta_i \leq 0$, then

$$\alpha_i^* = +\infty,$$

It is monotonic

Is the unique maximum of $\epsilon(\alpha_i)$ (convince yourselves that its derivative remains positive for all α_i). This means that removing the corresponding basis function from the model, increases the evidence.

RVM: Possible actions

The previous observations translate into three possible actions for each basis function i:

- **ADD**, if $\theta_i > 0$ and $\alpha_i = +\infty$:

$$\alpha_i^{\text{new}} = \frac{s_i^2}{\theta_i},$$
$$\Delta\mathcal{E} = \epsilon(\alpha_i^{\text{new}}).$$

- **RE-ESTIMATE**, if $\theta_i > 0$ and $\alpha_i < +\infty$:

$$\alpha_i^{\text{new}} = \frac{s_i^2}{\theta_i},$$
$$\Delta\mathcal{E} = \epsilon(\alpha_{\text{i}}^{\text{new}}) - \epsilon(\alpha_i).$$

- **DELETE**, if $\theta_i \leq 0$ and $\alpha_i < +\infty$:

$$\alpha_i^{\text{new}} = +\infty,$$
$$\Delta\mathcal{E} = -\epsilon(\alpha_i).$$

RVM: Updating the noise

Now, we assume that all the $\boldsymbol{\alpha}$'s are kept fixed and we attempt to maximize the evidence with respect to the inverse noise β.

Taking the derivative of the evidence with respect to β, you get:

$$\beta^{\text{new}} = \frac{N - M - \sum_i \alpha_i \Sigma_{ii}}{\| \boldsymbol{t} - \boldsymbol{\Phi}\boldsymbol{\mu} \|^2}.$$

This is a lengthy calculation. You are going to need the following formulas for a general matrix $\boldsymbol{A}$:

$$\frac{d\boldsymbol{A}^{-1}}{dt} = -\boldsymbol{A}^{-1}\frac{d\boldsymbol{A}}{dt}\boldsymbol{A}^{-1},$$

$$\frac{d|\boldsymbol{A}|}{dt} = |\boldsymbol{A}|\text{tr}\left[\boldsymbol{A}^{-1}\frac{d\boldsymbol{A}}{dt}\right].$$

RVM: Complete Algorithm

1. Initialize β and $\boldsymbol{\alpha}$ (Some of them might be $+\infty$. The corresponding basis functions are not in the model).

2. Compute: $\boldsymbol{\Sigma}, \boldsymbol{m}$ only for the basis functions in the model but **all** the s_i and h_i's.

3. For $i = 1, \ldots, M$, compute $\theta_i = h_i^2 - s_i$ and find the basis function i^* and the corresponding action $\mathcal{A}_{i^*}$ that gives the maximum increase in evidence.

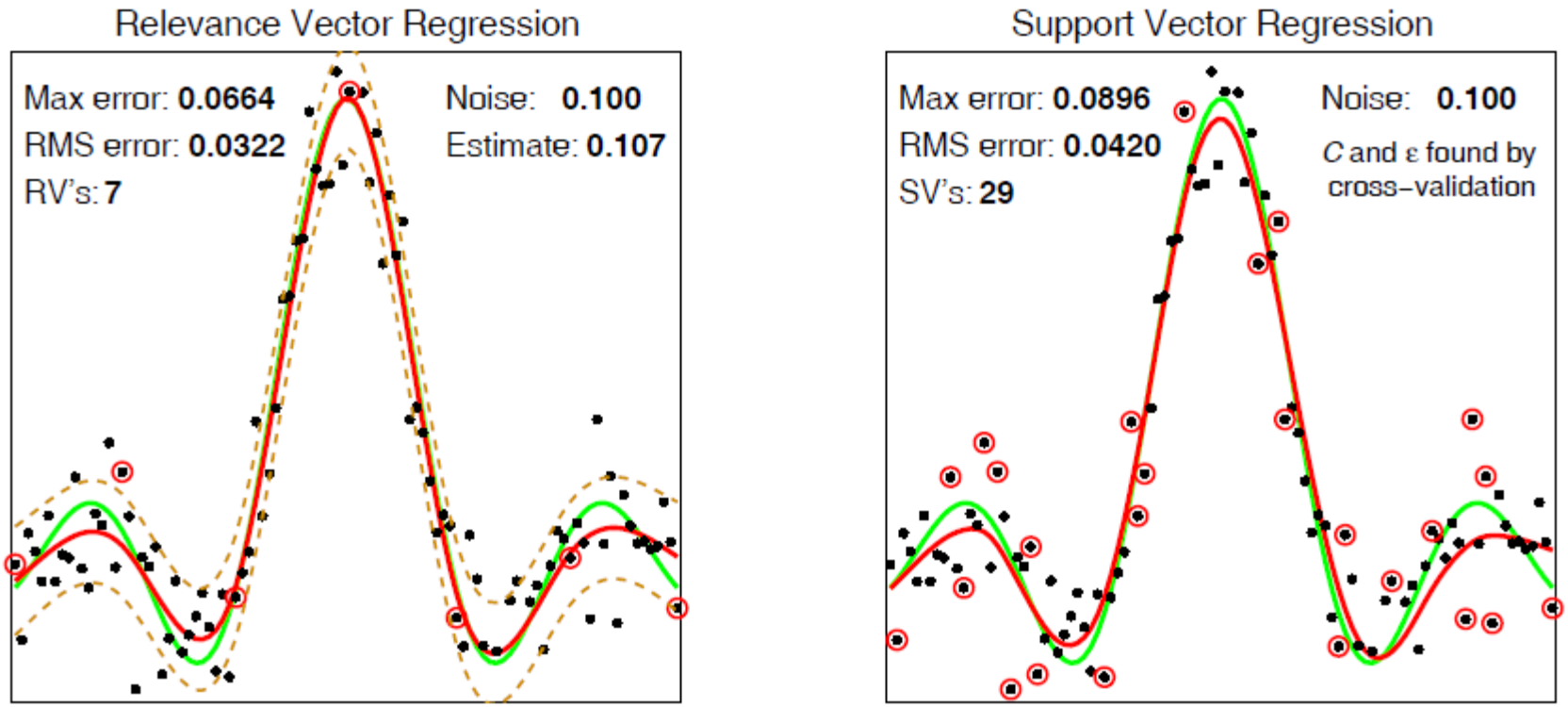
4. Perform the best action.

5. Re-estimate the noise β.

6. Re-compute (or update) $\boldsymbol{\Sigma}, \boldsymbol{m}, s_i$ and h_i.

7. If converged, stop. Otherwise go to 3.

A simple example



Fitting noisy samples from $f(x) = \frac{\sin(x)}{x}$. The basis functions are Gaussian kernels centered on the data points:

$$\phi_i(x) = k\left(x, x^{(i)}\right) = \exp\left(-\lambda\left(x - x^{(i)}\right)^2\right).$$

Taken from this very nice tutorial. The comparison is with SVM (a non-Bayesian competitor).

RVM: Fast updates

An efficient implementation of the RVM algorithm, requires a fast way to update the various statistics.

First, the mean and the co-variance matrix should only be build out of the **relevant** basis functions:

$$\boldsymbol{\Sigma}_r = \left(\text{diag}(\boldsymbol{\alpha}_r) + \beta \boldsymbol{\Phi}_r^{\text{T}} \boldsymbol{\Phi}_r\right)^{-1},$$
$$\boldsymbol{m}_r = \beta \boldsymbol{\Sigma}_r \boldsymbol{\Phi}_r^{\text{T}} \boldsymbol{y},$$

where the subscript "r" denotes that the corresponding quantity corresponds only to the basis functions in the model (the relevant basis functions), i.e. the ones with $\alpha_i < +\infty$. The other basis functions are simply irrelevant…

This requires some book-keeping. That is, at any iteration you must keep track of the set of relevant basis functions:

$$\mathcal{R} = \left\{ j \in \{1, \ldots, m\} : a_j < +\infty \right\}.$$

RVM: Fast updates

Secondly, we come to the statistics:

$$s_i = \boldsymbol{\varphi}_i^T \boldsymbol{C}_{-i}^{-1} \boldsymbol{\varphi}_i \quad \text{and} \quad h_i = \boldsymbol{\varphi}_i^T \boldsymbol{C}_{-i}^{-1} \boldsymbol{t}.$$

These are required for **all** basis functions, so that the change in evidence can be computed. It is more convenient to keep track of the following statistics instead:

$$S_i = \boldsymbol{\varphi}_i^T \boldsymbol{C}^{-1} \boldsymbol{\varphi}_i \quad \text{and} \quad \mathrm{H}_i = \boldsymbol{\varphi}_i^T \boldsymbol{C}^{-1} \boldsymbol{t}.$$

It is trivial to show that they connect with the original by:

$$s_i = \frac{\alpha_i S_i}{\alpha_i - S_i} \quad \text{and} \quad h_i = \frac{\alpha_i H_i}{\alpha_i - S_i}.$$

Using the Woodbury identity once more, you can find an easy way to compute the `capital statistics':

$$S_i = \beta \parallel \boldsymbol{\varphi}_i \parallel^2 - \beta^2 \boldsymbol{\varphi}_i^T \boldsymbol{\Phi}_r \boldsymbol{\Sigma}_r \boldsymbol{\Phi}_r^T \boldsymbol{\varphi}_i,$$
$$H_i = \beta \boldsymbol{\varphi}_i^T \boldsymbol{t} - \beta^2 \boldsymbol{\varphi}_i^T \boldsymbol{\Phi}_r \boldsymbol{\Sigma}_r \boldsymbol{\Phi}_r^T \boldsymbol{t}.$$

RVM: Fast updates

Finally, each action can be carried out without computing the statistics from scratch! You can derive them by repeatedly applying the Woodbury identity. It will take a couple of pages…

Notation:

- M_r: current number of relevant basis functions
- $i \in \{1, \ldots, M\}$: index of basis function that is currently being updated.
- $j \in \{1, \ldots, M_r\}$: the index within the **relevant** basis set that corresponds to i.
- $t \in \{1, \ldots, M\}$: index that runs over all basis functions (in or out of the model).

RVM: Fast updates

ADD basis function i:

$$\Delta\mathcal{E}^{\text{new}} = \frac{1}{2}\left(\frac{Q_i^2 - S_i}{S_i} + \log\frac{S_i}{Q_i^2}\right),$$

$$\boldsymbol{\Sigma}_r^{\text{new}} = \begin{pmatrix} \boldsymbol{\Sigma}_r + \beta^2\Sigma_{ii}\boldsymbol{\Sigma}_r\boldsymbol{\Phi}_r^T\boldsymbol{\varphi}_i\boldsymbol{\varphi}_i^T\boldsymbol{\Phi}_r\boldsymbol{\Sigma}_r & -\beta\Sigma_{ii}\boldsymbol{\Sigma}_r\boldsymbol{\Phi}_r^T\boldsymbol{\varphi}_i \\ -\beta\Sigma_{ii}\boldsymbol{\varphi}_i^T\boldsymbol{\Phi}_r\boldsymbol{\Sigma}_r & \boldsymbol{\Sigma}_{ii} \end{pmatrix},$$

$$\boldsymbol{m}_r^{\text{new}} = \begin{pmatrix} \boldsymbol{m}_r - \mu_i\beta\Sigma_{ii}\boldsymbol{\Sigma}_r\boldsymbol{\Phi}_r^T\boldsymbol{\varphi}_i \\ m_i \end{pmatrix},$$

$$S_t^{\text{new}} = S_t - \Sigma_{ii}(\beta\boldsymbol{\varphi}_t^T\boldsymbol{e}_i)^2,$$

$$H_t^{\text{new}} = H_t - \mu_i\beta\boldsymbol{\varphi}_t^T\boldsymbol{e}_i,$$

where

$$\Sigma_{ii} = (\alpha_i^{\text{new}} - S_i)^{-1},$$

$$\mu_i = \Sigma_{ii}H_i,$$

and

$$\boldsymbol{e}_i = \boldsymbol{\varphi}_i - \beta\boldsymbol{\Phi}_r\boldsymbol{\Sigma}_r\boldsymbol{\Phi}_r^T\boldsymbol{\varphi}_i.$$

Finally, m_r is increased by one.

CAUTION: There is a typo in Tipping and Faul.

RVM: Fast updates

DELETE basis function j:

$$\Delta\mathcal{E}^{\text{new}} = \frac{1}{2}\left(\frac{Q_i^2 - S_i}{S_i} - \log\left(1 - \frac{S_i}{\alpha_i}\right)\right),$$

$$\boldsymbol{\Sigma}_r^{\text{new}} = \boldsymbol{\Sigma}_r - \Sigma_{jj}^{-1}\boldsymbol{\Sigma}_j\boldsymbol{\Sigma}_j^T,$$

$$\boldsymbol{m}_r^{\text{new}} = \boldsymbol{m}_r - \mu_j\Sigma_{jj}^{-1}\boldsymbol{\Sigma}_j,$$

$$S_t^{\text{new}} = S_t + \Sigma_{\text{jj}}^{-1}\left(\beta\boldsymbol{\Sigma}_j^T\boldsymbol{\Phi}_r^T\boldsymbol{\varphi}_t\right)^2,$$

$$H_t^{\text{new}} = H_t + m_j\Sigma_{jj}^{-1}\beta\boldsymbol{\Sigma}_j^T\boldsymbol{\Phi}_r^T\boldsymbol{\varphi}_t,$$

where Σ_{jj} is the (j,j) element and $\boldsymbol{\Sigma}_j$ the j-th column of $\boldsymbol{\Sigma}_r$, resp., and m_j is the j-th element of $\boldsymbol{m}_r$.

After completing this step the corresponding rows and columns of the mean and the covariance must be removed.

The indices of the relevant vectors must be updated accordingly.

RVM: Fast updates

RE-ESTIMATE basis function j:

$$\Delta\mathcal{E}^{\text{new}} = \frac{1}{2}\left(\frac{Q_i^2}{S_i + \left(\left(\alpha_i^{\text{new}}\right)^{-1} - \alpha_i^{-1}\right)^{-1}} - \log\left(1 - S_i\left(\left(\alpha_i^{\text{new}}\right)^{-1} - \alpha_i^{-1}\right)\right)\right),$$

$$\boldsymbol{\Sigma}_r^{\text{new}} = \boldsymbol{\Sigma}_r - \kappa_j \boldsymbol{\Sigma}_j \boldsymbol{\Sigma}_j^T,$$

$$\boldsymbol{m}_r^{\text{new}} = \boldsymbol{m}_r - \kappa_j m_j \boldsymbol{\Sigma}_j,$$

$$S_t^{\text{new}} = S_t + \kappa_j\left(\beta \boldsymbol{\Sigma}_j^T \boldsymbol{\Phi}_r^T \boldsymbol{\varphi}_t\right)^2,$$

$$H_t^{\text{new}} = H_t + \kappa_j m_j \beta \boldsymbol{\Sigma}_j^T \boldsymbol{\Phi}_r^T \boldsymbol{\varphi}_t,$$

$\boldsymbol{\Sigma}_j$ the j-th column of $\boldsymbol{\Sigma}_r$, resp., and μ_j is the j-th element of $\boldsymbol{m}_r$ and

$$\kappa_j = \left(\Sigma_{jj} + \left(\left(\alpha_i^{\text{new}}\right)^{-1} - \alpha_i^{-1}\right)^{-1}\right)^{-1},$$

where Σ_{jj} is the (j,j) element.

RVM: Extension to multiple outputs

- We assume that we observe the data:

$$\mathcal{D} = \left\{\boldsymbol{x}^{(i)}, \boldsymbol{t}^{(i)} = \boldsymbol{f}\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{n},$$

and that we wish to learn the function $\boldsymbol{f}(\cdot) \in \boldsymbol{R}^q$.

- Pick a set of M basis functions $\boldsymbol{\phi}: \boldsymbol{R}^k \to \boldsymbol{R}^M$:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \left(\phi_1(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x})\right).$$

- We will approximate $\boldsymbol{f}(\cdot)$ by:

$$\hat{\boldsymbol{f}}(\boldsymbol{x}; \boldsymbol{W}) = \boldsymbol{W}^T \boldsymbol{\phi}(\boldsymbol{x}) + \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon}$ is a random vector representing noise and $\boldsymbol{W} \in \boldsymbol{R}^{M \times q}$.

- For simplicity, we choose to work with Gaussian noise with zero mean and inverse variance β, so the **likelihood** is:

$$p(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{W}, \beta) = \mathcal{N}_q(\boldsymbol{t}|\boldsymbol{W}^T \boldsymbol{\phi}(\boldsymbol{x}), \beta^{-1}).$$

RVM: Extension to multiple outputs

- The **prior on the weights**:

$$p(\boldsymbol{W}|\boldsymbol{\alpha}) = \prod_{j=1}^{M} p\left(\boldsymbol{w}_j|\alpha_j\right),$$

with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)$ and:

$$p\left(\boldsymbol{w}_j|\alpha_j\right) = \mathcal{N}_q\left(\boldsymbol{w}_j|0, \alpha_j^{-1}\right).$$

- The **posterior of the weights** is:

$$p(\boldsymbol{W}|\mathcal{D}, \beta, \boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{W}|\boldsymbol{M}, \boldsymbol{\Sigma}),$$

where:

$$\boldsymbol{\Sigma} = \left(\text{diag}(\boldsymbol{\alpha}) + \beta\boldsymbol{\Phi}^{\text{T}}\boldsymbol{\Phi}\right)^{-1},$$
$$\boldsymbol{M} = \beta\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\text{T}}\boldsymbol{Y},$$

with $\boldsymbol{Y} = \left(\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(N)}\right) \in \boldsymbol{R}^{N\times q}$ is the *vector of observations*, $\boldsymbol{\Phi} \in \boldsymbol{R}^{N\times M}$ is the *design matrix*:

RVM: Predictive distribution

- The **predictive** distribution is easily found as follows:

$$p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{\alpha},\beta) = \int p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{W},\beta)p(\boldsymbol{W}|\boldsymbol{\alpha})d\boldsymbol{W}$$
$$= \mathcal{N}(\boldsymbol{y}|\boldsymbol{M}^T\boldsymbol{\phi}(\boldsymbol{x}),\boldsymbol{\varphi}(\boldsymbol{x})^T\boldsymbol{\Sigma}\boldsymbol{\varphi}(\boldsymbol{x}) + \beta^{-1}).$$

So, we have the *predictive mean*:

$$\boldsymbol{\mu}(\boldsymbol{x}) = \boldsymbol{M}^T\boldsymbol{\phi}(\boldsymbol{x}),$$

and the predictive variance:

$$\sigma^2(\boldsymbol{x}) = \boldsymbol{\varphi}(\boldsymbol{x})^T\boldsymbol{\Sigma}\boldsymbol{\varphi}(\boldsymbol{x}) + \beta^{-1}.$$

Proof:

Complete the square….

End of proof.

RVM: Extension to multiple outputs

- **REMARK:** You might have to work with a scaled version of the data so that the prior assumptions are closer to reality. You may attempt to fit the function:

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{L}_{\text{obs}}^{-1}(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{m}_{\text{obs}}),$$

where $\boldsymbol{m}_{\text{obs}}$ is the *empirical mean of the data:*

$$\boldsymbol{m}_{\text{obs}} = \frac{1}{n}\sum_{i=1}^{N} \boldsymbol{t}^{(i)},$$

and $\boldsymbol{L}_{\text{obs}} \in \boldsymbol{R}^{q\times q}$ is the Cholesky decomposition of the *empirical covariance matrix of the data:*

$$\boldsymbol{C}_{\text{obs}} = \frac{1}{N-1}\sum_{i=1}^{N}\left(\boldsymbol{t}^{(i)} - \boldsymbol{m}_{\text{obs}}\right)\left(\boldsymbol{t}^{(i)} - \boldsymbol{m}_{\text{obs}}\right)^{T}.$$

If the empirical covariance is rank-deficient, you may use either an incomplete Cholesky or the eigen-decomposition. This is essentially a sort of output dimensionality reduction.

RVM: Extension to multiple outputs

- We proceed again, by maximizing the total evidence.
- Working out the details, it is easy to show that the total evidence is the sum of the evidence of each output:

$$\mathcal{E}(\boldsymbol{\alpha}, \beta | \mathcal{D}) = \sum_{j=1}^{q} \mathcal{E}\left(\boldsymbol{\alpha}, \beta \middle| \mathcal{D}_j\right),$$

where $\mathcal{D}_j = \left\{\left(\boldsymbol{x}^{(i)}, t_j^{(i)}\right)\right\}_{i=1}^{N}$.

- The algorithmic details remain essentially the same. The only difference is that:

$$H_i = \frac{1}{q}\sum_{j=1}^{q} H_{ji},$$

$$H_{ji} = \beta \boldsymbol{\varphi}_i^T \boldsymbol{t}_j - \beta^2 \boldsymbol{\varphi}_i^T \boldsymbol{\Phi}_r \boldsymbol{\Sigma}_r \boldsymbol{\Phi}_r^T \boldsymbol{t}_j.$$

RVM: Numerical stability concerns

- There are several numerical instabilities that should concern you.
- This is very common.
- For a numerically stable way to implement RVM using the Generalized Singular Value Decomposition look at Appendix D of I. Bilionis and N. Zabaras. *Multidimensional adaptive relevance vector machines for uncertainty quantification*. SIAM Journal for Scientific Computing, Vol. 34, No. 6, pp. B881–B908 2012.

RVM For Classification

- We can extend the relevance vector machine framework to classification problems by applying the ARD prior over weights to a probabilistic linear classification model.
- Consider two-class problems with a binary target variable $t \in \{0, 1\}$.
- The model now takes the form of a linear combination of basis functions transformed by a logistic sigmoid function

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sigma(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}))$$

- If we introduce a Gaussian prior over $\boldsymbol{w}$, then we obtain the model that has been considered in the logistic regression lecture.

RVM For Classification

- In the RVM, this model uses the ARD prior

$$p(w_j|\alpha_j) = \mathcal{N}(w_j|0, \alpha_j^{-1}).$$

- We can no longer integrate analytically over $\boldsymbol{w}$. Here we follow Tipping (2001) and use the Laplace approximation.

- We begin by initializing $\boldsymbol{\alpha}$. For this given value of $\boldsymbol{\alpha}$, we then build a Gaussian approximation to the posterior distribution and thereby obtain an approximation to the marginal likelihood ($\boldsymbol{A} = diag(\boldsymbol{\alpha_i})$)

$$lnp(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\alpha}) = ln\{p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\alpha)\} - lnp(\boldsymbol{t}|\boldsymbol{\alpha})$$

$$= \sum_{n=1}^{N} \{t_n lny_n + (1 - t_n)\ln(1 - y_n)\} - \frac{1}{2}\boldsymbol{w}^T \boldsymbol{A}\boldsymbol{w} + const.$$

- Maximization of this approximate marginal likelihood then leads to a re-estimated value for $\boldsymbol{\alpha}$, and the process is repeated until convergence.

RVM For Classification

- This can be done using iterative reweighted least squares (IRLS).
- We need the gradient and Hessian of the log posterior distribution

$$\nabla \ln p(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha}) = \boldsymbol{\Phi}^T(\boldsymbol{t} - \boldsymbol{y}) - \boldsymbol{A}\boldsymbol{w}$$
$$\nabla\nabla \ln p(\boldsymbol{w}|\boldsymbol{t},\boldsymbol{\alpha}) = -(\boldsymbol{\Phi}^T\boldsymbol{B}\boldsymbol{\Phi} + \boldsymbol{A})$$

- $\boldsymbol{B}$ is an $N \times N$ diagonal matrix with elements $b_n = y_n(1 - y_n)$, the vector $\boldsymbol{y} = (y_1, \ldots, y_N)^T$, and $\boldsymbol{\Phi}$ is the design matrix with elements $\Phi_{ni} = \phi_i(\boldsymbol{x}_n)$.
- At convergence of the IRLS algorithm, the negative Hessian represents the inverse covariance matrix for the Gaussian approximation to the posterior distribution.
- The mean & covariance of the Laplace approximation take the form

$$\boldsymbol{w} = \boldsymbol{A}^{-1}\boldsymbol{\Phi}^T(\boldsymbol{t} - \boldsymbol{y})$$
$$\boldsymbol{\Sigma} = (\boldsymbol{\Phi}^T\boldsymbol{B}\boldsymbol{\Phi} + \boldsymbol{A})^{-1}$$

RVM For Classification

- The marginal likelihood can be approximated as:

$$p(\boldsymbol{t}|\boldsymbol{\alpha}) = \int p(\boldsymbol{t}|\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\alpha})d\boldsymbol{w} \approx p(\boldsymbol{t}|\boldsymbol{w}^*)p(\boldsymbol{w}^*|\boldsymbol{\alpha})(2\pi)^{M/2}|\boldsymbol{\Sigma}|^{1/2}$$

- If we substitute for $p(\boldsymbol{t}|\boldsymbol{w}^*)$ and $p(\boldsymbol{w}^*|\boldsymbol{\alpha})$ and then set the derivative wrt α_i equal to zero:

$$-\frac{1}{2}(w_i^*)^2 + \frac{1}{2\alpha_i} - \frac{1}{2}\Sigma_{ii} = 0$$

- Defining $\gamma_i = 1 - \alpha_i\Sigma_{ii}$, gives:

$$\alpha_i^{new} = \frac{\gamma_i}{\left(w_i^*\right)^2}$$

- With $\hat{\boldsymbol{t}} = \boldsymbol{\Phi}\boldsymbol{w}^* + \boldsymbol{B}^{-1}(\boldsymbol{t} - \boldsymbol{y})$, we can write

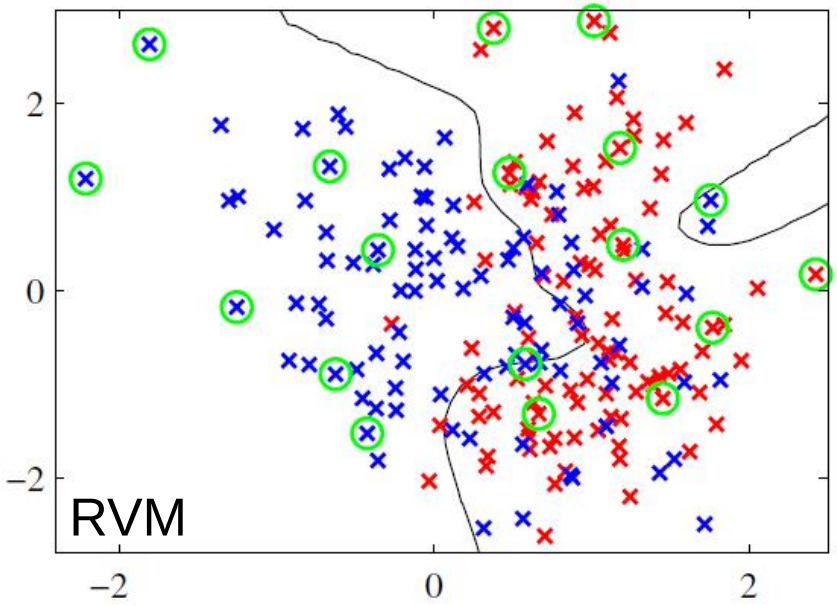
$$lnp(\boldsymbol{t}|\boldsymbol{\alpha}) = -\frac{1}{2}\{Nln(2\pi) + \ln|\boldsymbol{C}| + (\hat{\boldsymbol{t}})^T\boldsymbol{C}^{-1}\hat{\boldsymbol{t}}\},\ \boldsymbol{C} = \boldsymbol{B} + \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Phi}^T$$

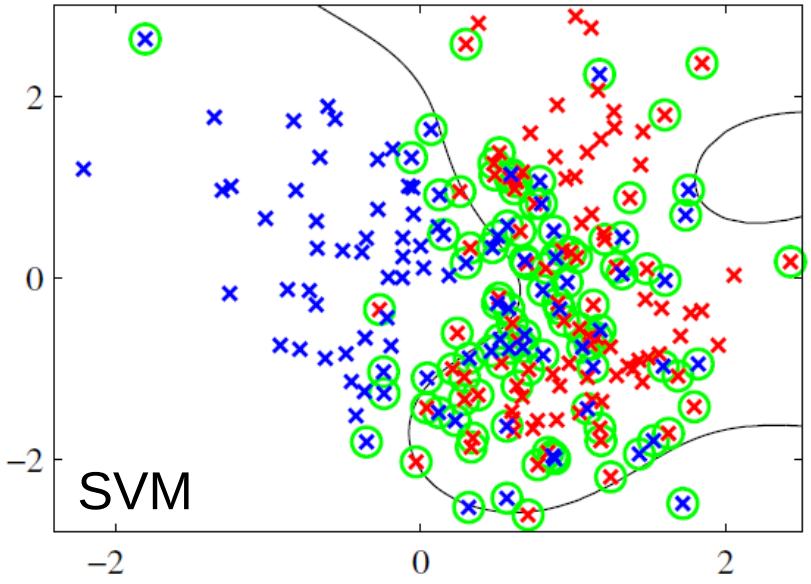
- This is the same form as for regression. Can apply the same sparsity analysis and obtain the same fast learning algorithm in which we optimize a single α_i at each step.

RVM For Classification

- Example of the RVM applied to a synthetic data set, in which the left-hand plot shows the decision boundary and the data points, with the relevance vectors indicated by circles.

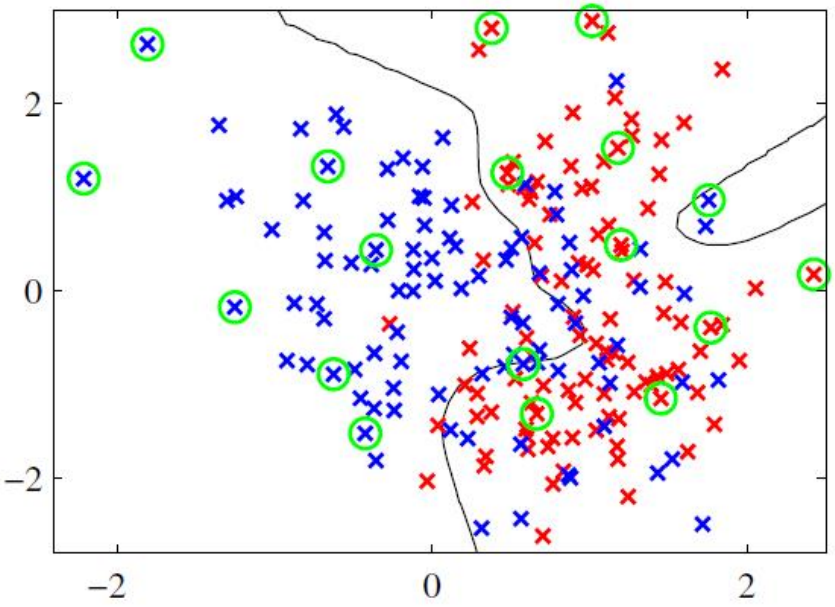
- Comparing with the SVM results it is seen that the RVM gives a much sparser model.

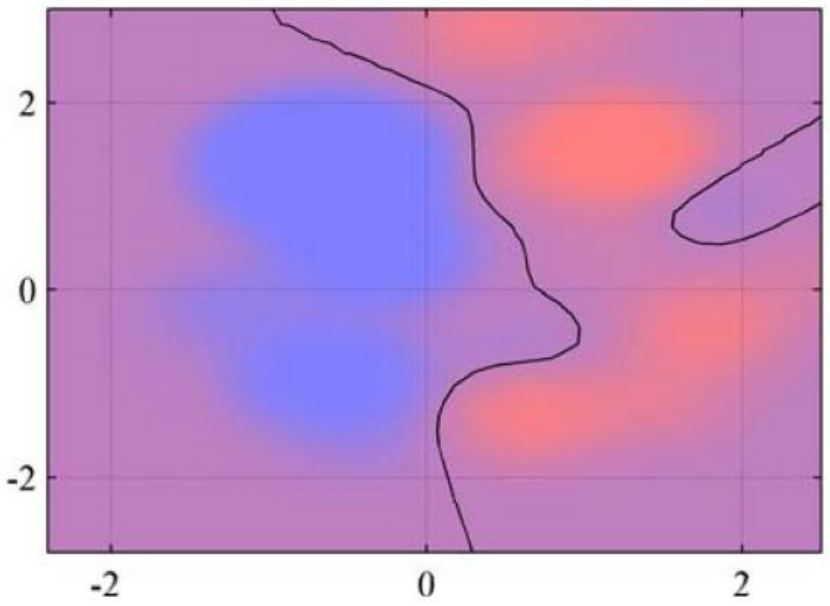




RVM For Classification

- The right-hand plot shows the posterior probability given by the RVM output in which the proportion of red (blue) ink indicates the probability of that point belonging to the red (blue) class.





RVM For Classification

- So far, we have considered the RVM for binary classification problems. For $K > 2$ classes,

$$y_k(\boldsymbol{x}) = \frac{\exp(\alpha_k)}{\sum_j \exp(\alpha_k)}$$

- The log likelihood function is then given by

$$lnp(\boldsymbol{T}|\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

where the target values t_{nk} have a 1-of-K coding for each data point n, and $\boldsymbol{T}$ is a matrix with elements t_{nk}.

- Again, the Laplace approximation can be used to optimize the hyperparameters in which the model and its Hessian are found using IRLS.

RVM For Classification

- The principal disadvantage is that the Hessian matrix has size $MK \times MK$, where M is the number of active basis functions, which gives an additional factor of K^3 in the computational cost of training compared with the two-class RVM.

- The principal disadvantage of the relevance vector machine is the relatively long training times compared with the SVM.

  - ✓ This is offset, however, by the avoidance of cross-validation runs to set the model complexity parameters.

  - ✓ Furthermore, because it yields sparser models, the computation time on test points, which is usually the more important consideration in practice, is typically much less.

References

- Scholkopf, B., A. Smola, R. C.Williamson, and P. L. Bartlett (2000). New support vector algorithms. *Neural Computation* **12**(5), 1207–1245.
- Valiant, L. G. (1984). A theory of the learnable. *Communications of the Association for Computing Machinery* **27**, 1134–1142.
- Anthony, M. and N. Biggs (1992). *An Introduction to Computational Learning Theory*. Cambridge University Press.
- Kearns, M. J. and U. V. Vazirani (1994). *An Introduction to Computational Learning Theory*. MIT Press.
- Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer.
- Vapnik, V. N. (1998). *Statistical learning theory*.Wiley.
- McAllester, D. A. (2003). PAC-Bayesian stochastic model selection. *Machine Learning* **51**(1), 5–21.

References

- T. Fletcher, Relevance Vector Machines Explained, October 2010.
- Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *J. of Machine Learning Research 1*, 211–244.
- MacKay, D. (1995b). Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. *Network*.
- Neal, R. (1996). *Bayesian learning for neural networks*. Springer.
- Bishop, C. and M. Tipping (2000). Variational relevance vector machines. In *UAI*.
- Buntine, W. and A. Weigend (1991). Bayesian backpropagation. *Complex Systems 5*, 603–643
- Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *J. of Machine Learning Research 1*, 211–244.
- Wipf, D. and S. Nagarajan (2010, April). Iterative Reweighted ℓ−1 and ℓ−2 Methods for Finding Sparse Solutions. *J. of Selected Topics in Signal Processing (Special Issue on Compressive Sensing) 4*(2).
- Wipf, D. and S. Nagarajan (2010, April). Iterative Reweighted ℓ−1 and ℓ−2 Methods for Finding Sparse Solutions. *J. of Selected Topics in Signal Processing (Special Issue on Compressive Sensing) 4*(2).
- Wipf, D. and S. Nagarajan (2007). A new view of automatic relevancy determination. In *NIPS*.
- MacKay, D. (1999). Comparision of approximate methods for handling hyperparameters. *Neural Computation 11*(5), 1035–1068.
- Wipf, D. and S. Nagarajan (2007). A new view of automatic relevancy determination. In *NIPS*
- Wipf, D. and S. Nagarajan (2010, April). Iterative Reweighted ℓ−1 and ℓ−2 Methods for Finding Sparse Solutions. *J. of Selected Topics in Signal Processing (Special Issue on Compressive Sensing) 4*(2).

References

- Wipf, D. and S. Nagarajan (2007). A new view of automatic relevancy determination. In *NIPS*.
- Rasmussen, C. E. and J. Qui˜nonero-Candela (2005). Healing the relevance vector machine by augmentation. In L. D. Raedt and S. Wrobel (Eds.), *Proceedings of the 22nd International Conference on Machine Learning*, pp. 689–696.
- T. Fletcher, Relevance Vector Machines Explained, October 2010.
- Faul, A. C. and M. E. Tipping (2002). Analysis of sparse Bayesian learning. In T. G. Dietterich, S. Becker, and Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems*, Volume 14, pp. 383–389. MIT Press.
- Tipping, M. E. and A. Faul (2003). Fast marginal likelihood maximization for sparse Bayesian models. In C. M. Bishop and B. Frey (Eds.), *Proceedings Ninth International Workshop on Artificial Intelligence and Statistics, Key West, Florida*.
- Williams, O., A. Blake, and R. Cipolla (2005). Sparse Bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(8), 1292–1304.